A Neurophysiologically-Inspired Statistical Language Model

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By
Jon Dehdari

Graduate Program in Linguistics The Ohio State University 2014

Dissertation Committee:
Professor William Schuler, Advisor
Professor Eric Fosler-Lussier
Professor Per Sederberg
© Jon Dehdari, 2014
Abstract

We describe a statistical language model having components that are inspired by electrophysiological activities in the brain. These components correspond to important language-relevant event-related potentials measured using electroencephalography. We relate neural signals involved in local- and long-distance grammatical processing, as well as local- and long-distance lexical processing to statistical language models that are scalable, cross-linguistic, and incremental. We develop a novel language model component that unifies n-gram, skip, and trigger language models into a generalized model inspired by the long-distance lexical event-related potential (N400). We evaluate this model in textual and speech recognition experiments, showing consistent improvements over 4-gram modified Kneser-Ney language models (Chen and Goodman, 1998) for large-scale textual datasets in English, Arabic, Croatian, and Hungarian.
Acknowledgements

I would like to thank my advisor, William Schuler, for his guidance these past few years. His easygoing attitude combined with his passion for his research interests helped steer me towards the topic of this dissertation. I’m also grateful for helpful feedback from Eric Fosler-Lussier and Per Sederberg in improving this work. Eric also provided much-appreciated guidance during my candidacy exams.

I’d like to thank my past advisors, Chris Brew, Detmar Meurers, and Deryle Lonsdale for shaping who I am today, and for their encouragement and instruction. I’m also grateful for insightful advice from Simon Dennis, Mike White, Eric Ringger, Josef van Genabith, Marie-Catherine de Marneffe, and Micha Elsner. The current and previous department chairs, Shari Speer and Beth Hume, have graciously gone out of their way for me. The department staff is the best I’ve seen, and their daily service has not gone unnoticed. Thank you Julie McGory, Julia Papke, Claudia Morettini, Joanna Anderson, Jane Harper, and Hope Dawson. I’ve also enjoyed years of working, both personally and professionally, with Jim Harmon. Thanks, Jimbo.

I’m grateful for interesting discussions with my fellow CL grad students, including Andy Plummer, Dennis Mehay, Preethi Jyothi, Raja Rajkumar, Steve Boxwell, DJ Hovermale, Kirk Baker, Yanzhang (Ryan) He, Dominic Espinosa, Adrianne Boyd, Ilana Heintz, Anton Rytting, Jianguo Li, David Howcroft, Marten van Schijndel, and Evan Jaffe. Many other people in the department have provided engaging and thought-provoking discussions, including (in alphabetical order) Alec Buchner, Zack
De, Manjuan Duan, Anouschka Foltz, Jirka Hana, Brian Joseph, Gregory Kierstead, Bob Levine, Scott Martin, Dan Miles, Jane Mitsch, Ila Nagar, Mike Phelan, Nathan Rasmussen, Pat Reidy, Helena Riha, Judith Tonhauser, Rory Turnbull, Na’im Tyson, Abby Walker, Kodi Weatherholtz, Don Winford, Chris Worth, and Murat Yasavul.

I would like to thank all those at DCU as well, including Lamia Tounsi, Josef van Genabith, as well as the following (in alphabetical order): Mohammed Attia, Özlem Çetinoğlu, Sandipan Dandapat, Sarah Ebling, Maria Eskevich, Mikel Forcada, Jennifer Foster, Yvette Graham, Teresa Lynn, Montse Maritxalar, Sergio Penkale, Robert Smith, Ankit Srivastava, Antonio Toral, and Joachim Wagner.

I’m also grateful for all those who have contributed to Free software and Free content, which has made this research and dissertation possible.

I’d like to thank my seventh-grade English teacher for telling me I didn’t know English good enough to learn a foreign language!

I’m very thankful for a loving, supportive, and patient family. My parents fostered an appreciation for learning and education. All of my extended and immediate family have helped me every step of the way. Gwen has kept me young. She continually reminds me what a wonderful and amazing world this is. My greatest appreciation goes to Carolyn. I’m thankful for her unfailing love and encouragement.
Vita

2004 ........................................ Bachelor of Arts, Linguistics
                   Minor in Arabic
                   Brigham Young University
2006 ........................................ Master of Arts, (Computational) Linguistics
                   Brigham Young University

Publications

Mehay, Dennis N., and Jon Dehdari. 2012. The Ohio State University Farsi-English
System Submission to OpenMT 2012. In Proceedings of the NIST 2012 Open
Machine Translation Workshop (OpenMT 2012), Washington D.C., USA.

Dehdari, Jon, Lamia Tounsi, and Josef van Genabith. 2011. Morphological Features
for Parsing Morphologically-rich Languages: A Case of Arabic. In Proceedings of
the Second Workshop on Statistical Parsing of Morphologically Rich Languages,

Jadidinejad, Amir Hossein, Fariborz Mahmoudi, and Jon Dehdari. 2010. Evaluation
of Perstem: A Simple and Efficient Stemming Algorithm for Persian. In Peters,
C., Nunzio, G. D., Kurimo, M., Mandl, T., Mostefa, D., Peñas, A., and Roda,
G., editors, Multilingual Information Access Evaluation I. Text Retrieval Experi-
ments, volume 6241 of Lecture Notes in Computer Science, 98–101. Springer,
Heidelberg.

Preiss, Judita, Jon Dehdari, Josh King, and Dennis Mehay. 2009. Refining the
most frequent sense baseline. In Proceedings of the NAACL HLT Workshop on
Semantic Evaluations (SEW-2009), 10–18, Boulder, CO, USA. Association for
Computational Linguistics.
Fields of Study

Major Field: Linguistics

Statistical language modeling, machine translation, parsing, unsupervised learning, morphological analysis & stemming, speech recognition, Persian NLP, Arabic NLP, and efficient large-scale NLP systems.
Table of Contents

Abstract ................................................................. ii
Acknowledgements ......................................................... iii
Vita ................................................................. v
List of Figures ........................................................... ix
List of Tables ............................................................ xii
1 Introduction ........................................................... 1
2 Event-Related Potentials .............................................. 5
  2.1 ELAN .............................................................. 6
  2.2 N400 .............................................................. 6
  2.3 P600 .............................................................. 8
3 Statistical Language Models ......................................... 12
  3.1 Cache Language Models .......................................... 18
  3.2 Skip Language Models ............................................ 19
  3.3 Trigger Language Models ......................................... 21
  3.4 Topic-based Language Models .................................... 23
  3.5 Neural Network Language Models ................................ 26
    3.5.1 Feedforward Neural Networks .............................. 26
    3.5.2 Recurrent Neural Networks ................................ 30
  3.6 Maximum Entropy Language Models ............................. 35
  3.7 Summary .......................................................... 38
4 Description of the Neurophysiologically-inspired Language Model .... 40
  4.1 Overview .......................................................... 40
  4.2 ELAN-based Language Model Component ........................ 41
  4.3 N400-based Language Model Component ........................ 44
    4.3.1 Illustrations ................................................ 52
  4.4 P600-based Language Model Component ........................ 62
List of Figures

3.1 Bleu scores on NIST OpenMT 2006 Arabic-English test set by Brants et al. (2007), using either interpolated Kneser Ney smoothing (Chen and Goodman, 1998) or their unnormalized Stupid Backoff. ............ 16

3.2 Plate notation (cf. Buntine, 1994) of a topic-based language model, based on Gildea and Hofmann (1999, pg. 1). Here \( h \) stands for the word history, \( t \) for topic, and \( w \) for word. Hidden nodes are white, visible nodes shaded. ................................................................. 23

3.3 Dynamic Bayesian network (Dean and Kanazawa, 1989) of topic-based language model (Gildea and Hofmann, 1999). .......................... 24

3.4 Single hidden-layer feedforward neural network used by Bengio et al. (2003, from pg. 1142). .......................... 28

3.5 Recurrent neural network by Elman (1990, from pg. 4). ........... 31

3.6 Left: Elman network (Elman, 1990); Right: unfolded Elman network with \( \tau = 3 \). From Bodén (2002). .......................... 32

4.1 Product of two scaling factors (green) from \( e^{-\lambda_(i-j)} \) (yellow) and \( c'(w_j)^{-p_h} \) (blue), where \( c'(w_j) \) comes from the first 300 million words of the English Gigaword-4 corpus. ........ 48

4.2 Future probability mass allocation for \( P_d \) at ‘cooked’, with varying decay constants. ............................. 54
4.3 Dynamic Bayesian Network of a decaying language model’s $P_d$, using extended bigrams (top edges) and extended trigrams (bottom edges). Based on experimental results $\lambda_d = 0.4$ for both sets of edges in this illustration.

4.4 Dynamic Bayesian Network of an interpolated $n$-gram language model (Jelinek and Mercer, 1980), using bigrams (top edges) and trigrams (bottom edges).

4.5 Ranked probabilities of predicted words in the Switchboard corpus, given the history “BECAUSE I WENT THROUGH THE PUBLIC UM”.

Both the $x$ and $y$ axes are in a log$_{10}$ scale.

5.1 Perplexity and word error rate results of various language models on the Kennedy test set (Klakow and Peters, 2002, pg. 26)

5.2 Perplexity and word error rate results of various language models on the Eval96 and Eval97 test sets (Klakow and Peters, 2002, pg. 26)

5.3 Cross-entropy and word error rate results of various language models reported in Goodman (2001, pg. 38)

5.4 Plots of four test set function approximations, using linear scales. Solid line segments indicate attested ranges.

5.5 Interpolation weights for English, Arabic, Croatian, and Hungarian corpora, using $|C| = 200$ classes in the class-based model.

5.6 Interpolation weights for English, Arabic, Croatian, and Hungarian corpora, using $|C| = 400$ classes in the class-based model.

5.7 Interpolation weights for English, Arabic, Croatian, and Hungarian corpora, not using the class-based model.
5.8 Decay constants for English, Arabic, Croatian, and Hungarian corpora, using $|C| = 400$ classes. .................................................. 91

5.9 Decay constants for English, Arabic, Croatian, and Hungarian corpora, not using the class-based model. ................................. 92

5.10 Interpolation weights for English Switchboard corpus 1r2, with the first plot using $|C| = 200$ classes. The second plot is not using a class-based model. ......................................................... 99

A.1 Stylized representation of the action potential in a mammalian pyramidal neuron (courtesy of Storm, 1987; Bean, 2007). ................. 135

A.2 Surface of left hemisphere of the human brain (from Gray, 1918). . 136
List of Tables

3.1 Experimental perplexity results of Bengio et al. (2003)’s feedforward neural network with one hidden layer, trained on either 800K words of the Brown corpus or 14M words of an AP news corpus. . . . . . . . . . 30

3.2 Comparison of different language models. Distance: 1=short: no intervening words between history and predicted word; 2=medium: distance-dependent history with intervening words; 3=long: arbitrary dependencies/relations within sentence, 4=long across sentence boundaries. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

4.1 Examples from the Switchboard development set, with relative improvements of using a decaying language model over using a 4-gram modified Kneser-Ney model. The greyed text in the predicted word column is additional context for the reader (of this dissertation). . . . . . . . . 56

4.2 Top 40 predicted words given the history “BECAUSE I WENT THROUGH THE PUBLIC UM”, trained on the training section of the Switchboard corpus. Probabilities are in a log_{10} scale. . . . . . . . . . . . . . . . 58

4.3 Top 40 predicted words given the history “because i went through the public um””, trained on 100 million words of the English Gigaword-4 corpus. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

5.1 Pearson’s correlation coefficients of perplexity and word error rate on various test sets of Klakow and Peters (2002, pg. 27) . . . . . . . . . . . 74
5.2 Full training set sizes of various textual corpora. 81
5.3 Test set sizes of various textual corpora. 81
5.4 Perplexities of test sets using 4-gram modified Kneser-Ney models, as well as adding a $\nu\phi$-LM with $|C| = 200$ classes. 83
5.5 Perplexities of test sets using 4-gram modified Kneser-Ney models, as well as adding a $\nu\phi$-LM with $|C| = 400$ classes. 85
5.6 Perplexities of test sets using 4-gram modified Kneser-Ney models, as well as adding a decaying language model. 88
5.7 Switchboard transcription training, development, and test set sizes. 95
5.8 Perplexities of the Switchboard transcription test set using 4-gram modified Kneser-Ney models, as well as adding a $\nu\phi$-LM with $|C| = 200$ classes (top), or a decaying model (bottom). 97
5.9 Word error rates of the Switchboard development set using trigram models, as well as adding a $\nu\phi$-LM with $|C| = 200$ classes (top), or a decaying model (bottom). 98
5.10 Word error rates of the Switchboard test/eval set using trigram models, as well as adding a $\nu\phi$-LM with $|C| = 200$ classes (top), or a decaying model (bottom). 100
5.11 Word error rate percentages of the Switchboard development set using various lattices and 100-best lists. Here $AM$ refers to the acoustic model cost interpolation weight and $G$ refers to the decoding graph cost interpolation weight, which graph is the product of the language model cost, transition probabilities, and any pronunciation cost. The interpolation weights here need not sum to one. 104
6.1 Summary of the training corpora in our experiments. . . . . . . . 110

B.1 Labels for various joint and marginal distributions involving $w_1$ and $w_2$. Here $\neg w_1$ denotes $\{w| w \in V, w \neq w_1\}$, or the complement of $w_1$. 140

B.2 Various association measures and the joint and/or marginal distributions that they use. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142
Chapter 1

Introduction

Natural language is a distinctly human phenomenon, processed in the human brain. We describe a statistical language model in this paper that makes use of the wealth of research in how the brain processes natural language in real time to develop a model that seeks to accurately reflect macroscopic neurophysiological processes, on a functional level. In doing so, we aim to achieve better results in natural language processing tasks that make use of statistical language models. We adhere to a few fundamental principles in the design of the model.

First and foremost, our model reflects the multicomponent nature of language processing in the brain (cf. Hasson and Small, 2008). There are local and long-distance grammatical relations in a given utterance, as well as local and long-distance lexical relations.¹ These four different levels can be treated distinctly for greater specialization. This allows for more refined optimization strategies, as we are able to split the components into different feature functions. Functionally-monolithic neural net-based language models like Xu and Rudnicky (2000), Emami et al. (2003), Bengio et al. (2003), and Schwenk (2007, 2010) fail to capture the multiple levels of linguistic interactions among sentential constituents mentioned above.

¹Not to mention prosodic information.
Secondly, the reality of the computing landscape today is that personal computers no longer dominate like they used to. Computers have become both much smaller and much larger. Statistical language models need to be able to be trained on massive amounts of data, using parallel computing architectures such as MapReduce (Dean and Ghemawat, 2004). They also need to be able to be fast and lightweight, due to the fact that machine translation and speech recognition systems call upon the language model \textit{tens or hundreds of thousands of times per sentence}.\footnote{For example, Brants \textit{et al.} (2007) report querying 150,000 \textit{n}-grams per sentence. See Koehn (2010, pp. 210–211) for further discussion.}

Thirdly, there are over 5,000 languages in the world (Lewis, 2009). Within each language there are several domains, registers, dialects, etc. Manual annotation is not an option for the vast majority of the world’s languages, and even for many forms of major languages. Therefore we seek to design a statistical language model that uses unannotated input.

The non-biologically-based composite language model of Wang \textit{et al.} (2005, 2006) and Tan \textit{et al.} (2011, 2012) uses \textit{n}-gram models, structured language models (Chelba and Jelinek, 1998; Chelba, 2000), and probabilistic latent semantic analysis (cf. Gildea and Hofmann, 1999). They used a supervised PCFG parser to derive the structured language model \textit{sic}, which reduces the number of languages under consideration. They found that to use even a moderate amount of training data (230M tokens) on a cluster of hundreds of nodes they must limit the number of topics to just five due to the extensive training time required for PLSA. In order to train a 5-topic PLSA model on 1.3B tokens of the English Gigaword corpus (Parker \textit{et al.}, 2011b) it took 500

\footnote{For example, Brants \textit{et al.} (2007) report querying 150,000 \textit{n}-grams per sentence. See Koehn (2010, pp. 210–211) for further discussion.}
cluster nodes working for over 75 hours. Furthermore, this composite language model is not incremental, so they are required perform reranking of translation output.

Our work provides several substantial contributions. We assemble different language model components in a principled manner that account for various aspects of linguistic levels. We also discuss the neurophysiological basis for these different components. Each of the components, as well as the combined model, is incremental, cross-linguistic, and scalable. These three attributes are vitally important for any language model that aims to supplant the longstanding entrenched position of solely using n-gram language models (§3) in NLP systems.

Note that we are describing a generative, incremental statistical language model to be employed in natural language processing systems; we are not describing a cognitive architecture, nor are we introducing a new psycholinguistic theory. That is, we use neuroscientific research to guide us to build a better statistical language model, rather than implement a psychological theory of how the mind works or processes language.

In the next section we discuss the brain responses that pertain to language

---

3For example, Soar (Newell, 1990), ACT-R (Anderson and Lebiere, 1998), EPIC (Kieras and Meyer, 1997), LIDA (Snaider et al., 2011), 4CAPS (Just and Varma, 2007), Icarus (Langley et al., 2005), CHREST (Gobet and Lane, 2010), Clarion (Sun, 2006), DUAL (Kokinov, 1994), or Polyscheme (Cassimatis, 2002).

4Although there are some properties that our statistical language model shares with certain theories in psycholinguistics. Our language model is composite, consisting of multiple components that draw upon various sources of information, like constraint-based models do (Trueswell et al., 1993; Macdonald et al., 1994; McRae et al., 1998, inter alia). Due to limitations in computer memory, many machine translation and speech recognition systems decode using a beam search, which is a greedy search strategy. Thus the statistical language model is not used by these systems in the fully parallel manner of a typical constraint-based models, nor is backtracking (reanalysis) employed as in serial models (Frazier, 1979; Fodor and Frazier, 1980; Frazier, 1987; Ferreira and Henderson, 1990, inter alia). Also, in section 4.3 we show that our N400-based language model component is consistent not only with neurolinguistic findings, but also with psycholinguistic research in lexical processing.
processing, especially as they relate to developing a statistical language model. In section 3 we discuss $n$-gram language models and their shortcomings, then in section 4 we detail our model. Section 5 describes our experiments and analyzes the results, then we consider future directions in section 6.1.
Chapter 2

Event-Related Potentials

Event-related potentials (ERPs) are averaged fluctuations in voltage at various sites on the scalp, due to synaptic activity on the dendrites of cortical pyramidal cells (Stemmer and Whitaker, 1998). These are direct reflections of neural activity, providing very high precision of the timing of neural activity at the neocortex (Kutas et al., 2012). Most of the names of the ERP components reflect polarity (negative or positive voltage directions), manifest as either $N$ or $P$ in the name. Many also give an indication as to the average time in milliseconds of peaking after the onset of the stimulus. Some alternatively describe the location on the scalp where electroencephalography sensors detect the wave most often.

Over the years, some components of these event-related potentials came to be known to occur as a consequence of language-related stimuli. These are summarized below, with further explanation in subsequent sections.

The **ELAN** (early left-anterior negativity) relates to the local, incremental assembling of word categories. The **N400** involves the sentence-level integration of lexemes, given a lexical-semantic context. The **P600** is implicated in broad structural reanalysis, revision, long-distance agreement relations, and filler-gap processing. There are other event-related potential components involved in phonological processing (PMN),
prosodic boundary detection (CPS), as well as more general-purpose components (N100, N170, N200, CNV, MMN, P100, P200, P3a, P3b, LPC).

2.1 ELAN

First reported by Brown \textit{et al.} (1979) and extensively covered by Angela D. Friederici (1993; 1996; 1999, \textit{inter alia}), left-lateralized anterior negativity occurs 100–300 ms after the onset of certain stimuli. Early left-anterior negativity (ELAN) relates to local, automatic, first-pass morpho-syntactic violations, but not longer-distance syntactic errors (Neville \textit{et al.}, 1991; Frisch \textit{et al.}, 2004; Steinhauer and Connolly, 2008; Friederici, 2002). Friederici (2002) argues that the temporal phase occurring during ELAN-related processing primarily involves the local, incremental assembling of word categories.

2.2 N400

First reported in Kutas and Hillyard (1980), the N400 is elicited in situations involving lexical semantics. Specifically it involves the association between content words in a sentence context and the target word, rather than whole-sentence truth-conditional semantics (Fischler \textit{et al.}, 1983; Kuperberg \textit{et al.}, 2003). For example, these authors found N400 negativity with the sentence “a sparrow is not a vehicle”.\footnote{Fischler \textit{et al.} (1983) controlled for the fact that negated sentence requires extra processing.} This sentence is true in terms of the truth-conditional semantics, but contains content words that are not typically associated with each other.
Kutas and Hillyard (1988) shed more light on the N400, finding that words at the beginning of a sentence elicit larger N400 than those at the end of the sentence, because of the additional contextual constraints given in the latter. That is, they found a continuum of N400 negativity that is inversely proportional to the amount of sentence context.

Van Petten (1993) reported that sentential context is built-up incrementally as each word is given. She also noted that the frequency of the particular content words in the sentence context helps determine the magnitude of N400 effects (see also Van Petten and Kutas, 1990, 1991).

Van Petten and Kutas also found other interesting characteristics of the N400 component, which can help influence the design of a lexical-semantic component in a statistical language model. They found that the N400 is not sensitive to syntactic constraints (cf. Kutas and Hillyard, 1983; Van Petten and Kutas, 1991; Neville et al., 1991). They also found that contexts containing only closed-class words did not facilitate the processing of subsequent open-class words (Van Petten and Kutas, 1991). Thus, we should ensure that function words play a minimal role in predicting content words. This might suggest a dynamic weighting scheme, where the lexical-semantic component is weighted more when the class-based sequence model predicts an open-class word, and lower when the class-based model predicts closed-class words.

Van Petten and Kutas also found that syntactic constraints were applied locally (Van Petten and Kutas, 1991, p. 109), using only the immediately preceding word or two to process closed-class words. They contrasted this with the cumulative nature of sentence context for meaning-rich open-class words.
2.3 P600

Osterhout and Holcomb (1992) first discussed a centroparietal positivity starting around 500 ms post-onset, with a maximum polarity around 600 ms. Whereas ELAN has been linked with early, first-pass, and localized parsing, the P600 has been explained in terms of broader structural reanalysis and revision (Friederici et al., 1996; Steinhauer and Connolly, 2008). For example, Neville et al. (1991) and McKinnon and Osterhout (1996) found P600 in subadjacency/ECP violations, while Osterhout and Mobley (1995) found it in a variety of agreement violations. Such agreement violations included number between the subject and the verb, number between a reflexive pronoun and its antecedent, as well as grammatical gender between a reflexive pronoun and its antecedent.

Verb argument structure violations also elicit a P600 deflection (Friederici and Frisch, 2000). Friederici and Frisch used intransitive verbs in a sentence containing a subject and an object, such as:

(1) *Anna weiß, dass der Kommissar den Banker abreiste und ...  
*Anna knows that the.NOM inspector the.ACC banker departed and ...  
"*Anna knows that the inspector departed the banker and ..."

Not only did they find a P600 for verb argument structure violations involving incorrect number of arguments, they also found a P600 when the type of argument was ungrammatical, such as an accusative article instead of a dative:

(2) *Anna weiß, dass der Kommissar den Banker beistand und ...  
Anna knows that the.NOM inspector the.ACC banker helped and ...  
"Anna knows that the inspector helped the banker and ..."
Friederici and Frisch found similar results for sentences having verb-second placement as well. These types of sentences have the verb preceding both the subject and the object.

Another source of the P600 involves the incremental processing of locally ambiguous sentences, especially for the non-dominant local parse (Osterhout and Holcomb, 1992; Mecklinger et al., 1995; Friederici et al., 1998; Steinhauer et al., 1999; Frisch et al., 2002; beim Graben et al., 2008, inter alia). Osterhout and Holcomb (1992) used the following sentence to obtain a P600:

(3) The broker persuaded to sell the stock was sent to jail.

where the preferred reading, given the first three words suggests that persuaded and the subsequent words will be part of the predicate, rather than a reduced relative clause.

The P600 is not restricted to ungrammatical or locally-ambiguous sentences. Kaan et al. (2000) made the interesting finding that a posterior P600 is produced for sentences containing relatively long dependencies, including filler-gap constructions. For example, they obtained the greatest P600 in (4c), and the least in (4a):

(4) a. Emily wondered whether the performer in the concert imitated a pop star . . .
   b. Emily wondered who the performer in the concert imitated . . .
   c. Emily wondered which pop star the performer in the concert had imitated . . .

where the filler and gaps are underlined. Based on results for these types of sentences, as well as for ambiguous, but preferred, sentences, Frisch et al. (2002) and Kaan et al.
(2000) have argued that the P600 component is an indicator of syntactic processing cost.

Fiebach et al. (2001, 2002) confirmed the assertion that the P600 evinces syntactic processing, rather than working memory (cf. Gibson, 1991, 1998), by analyzing both short and long distance object WH-questions in German. They compared P600 deflections of short vs. long distance object WH-questions compared with short vs. long distance subject WH-questions, expecting to elicit a P600 for the object WH-questions where the filler stored in working memory can be integrated into the phrase structure. For example, sentences such as those found in (5) and (6) below were used, having a short- or long-distance subject or object WH-word:

(5) a. Thomas fragt sich, wer am Dienstag den Doktor verstündigt hat.
Thomas asks himself, who.NOM on Tuesday the.ACC doctor called has.
“Thomas wonders who called the doctor on Tuesday.”

b. Thomas fragt sich, wen am Dienstag der Doktor verstündigt hat.
Thomas asks himself, who.ACC on Tuesday the.NOM doctor called has.
“Thomas wonders who the doctor called on Tuesday.”

(6) a. Thomas fragt sich, wer am Dienstag nachmittag nach dem Unfall den Doktor verstündigt hat.
Thomas asks himself, who.NOM on Tuesday afternoon after the accident the.ACC doctor called has.
“Thomas wonders who called the doctor on Tuesday afternoon after the accident.”

b. Thomas fragt sich, wen am Dienstag nachmittag nach dem Unfall der Doktor verstündigt hat.
Thomas asks himself, who.ACC on Tuesday afternoon after the accident the.NOM doctor called has.
“Thomas wonders who the doctor called on Tuesday afternoon after the accident.”

Fiebach et al. indeed found a late centroparietal positivity as a result of object WH-question sentences, regardless of whether the objects were near or far from their gap position. A different ERP was implicated in the question of memory storage
costs. Thus a unifying attribute of the P600 is indexing the integration of syntactic information (Kutas et al., 2006), consistent with findings for English from Kaan et al. (2000) and Phillips et al. (2005).

\footnote{Namely, left anterior negativity. See King and Kutas (1995) and Phillips et al. (2005), \textit{inter alia}}
Chapter 3

Statistical Language Models

Many tasks in natural language processing, such as machine translation, speech recognition, optical character recognition, predictive text, and virtual keyboards, make use of an incremental statistical language model, which provides a probability of a particular word $w_i$ occurring given a history $h$ of previous words. The probability of the entire sentence is defined as:

$$P(w_1 \ldots w_k) = P(w_1)P(w_2|w_1) \ldots P(w_k|w_1, w_2, \ldots, w_{k-1})$$

$$= \prod_{i=1}^{k} P(w_i|w_1, \ldots, w_{i-1}, h)$$ (3.1)

It is common to make a Markov assumption, which replaces the entire sentence history $h$ with an approximation from the immediately-preceding word(s):

$$P(w_1 \ldots w_k) \approx \prod_{i=1}^{k} P(w_i|w_{i-1})$$ (3.2)

where the maximum likelihood estimate of $P(w_i|w_{i-1})$ is based on counts $c(\cdot)$

$$P_{ML}(w_i|w_{i-1}) \triangleq \frac{c(w_{i-1}w_i)}{\sum_{w \in W} c(w_{i-1}w)}$$ (3.3)
This model can be illustrated as a dynamic Bayesian network, where $w_0$ represents the start state $<s>$:

The Markov assumption is employed due to limitations in training set size. This approximation has implications that affect grammaticality and lexical choice. This limited, localized history restricts the languages that the model can generate.\(^1\) Furthermore, ignoring a word $w_j$ where $i - n > j$\(^2\) can also affect the overall lexical-semantic coherency of the sentence. Consider the example fragment

(1) I cooked the fish in the

If we increase the Markov order, so that $n = 3$ (as in Equation 3.4 and its corresponding dynamic Bayesian network below) it would still leave comparatively little probability to what we would intuitively be expecting the next word to be.

$$P(w_1 \ldots w_k) = \prod_{i=1}^{k} P(w_i|w_{i-2}, w_{i-1})$$  \hspace{1cm} (3.4)

Setting $n = 4$ can help in this regard since fish is now in its local history, but requires much more training data to prevent overfitting. Even at this high order, the

---

\(^1\)Namely, regular languages.

\(^2\)Where $n$ is the $n$-gram order, $w_i$ is the next word to be generated, and $w_j$ is a word in the full history $h$. 

13
model still fails to allocate probability to what we would intuitively expect, in light of the entire history in example (1). Given the first 300 million words of the English Gigaword-4 corpus, the words with the highest probability of following example (1) are: waters, area, pond, world, united, south, haricot, contested, water, us, troubled, ship, river, caspian, areas.

Given the 300 million word training set mentioned above, a simple estimation of $P(w_i|w_{i-1})$ like maximum likelihood estimation would give a probability of 0 to any culinary words immediately following example (1). We could backoff (Katz, 1987) to a lower order of $n$, using a smoothing algorithm of the general form (Chen and Goodman, 1998):

$$P_{BO}(w_i|w_{i-n+1}...w_{i-1}) \triangleq \begin{cases} 
\alpha(w_i|w_{i-n+1}...w_{i-1}) & \text{if } c(w_{i-n+1}...w_{i}) > 0 \\
\gamma(w_{i-n+1}...w_{i-1})P_{BO}(w_i|w_{i-n+2}...w_{i-1}) & \text{if } c(w_{i-n+1}...w_{i}) = 0
\end{cases}$$

In other words, if an $n$-gram occurs in the training set, we just modify the maximum likelihood estimate so that the conditional distribution still sums to one, giving $\alpha(\cdot)$. Otherwise we scale the next-lower order $n$-gram using $\gamma$ so that the conditional distribution sums to one.

An alternative way of smoothing is to interpolate $P(w_i|w_{i-n+1}...w_{i-1})$ and $P(w_i|w_{i-n+2}...w_{i-1})$ (Jelinek and Mercer, 1980):

$$P_{JM}(w_i|w_{i-n+1}...w_{i-1}) \triangleq \lambda_{w_{i-n+1}...w_{i-1}}P_{ML}(w_i|w_{i-n+1}...w_{i-1}) + (1 - \lambda_{w_{i-n+1}...w_{i-1}})P_{JM}(w_i|w_{i-n+2}...w_{i-1})$$
where \( \lambda_{w_{i-n+1}...w_{i-1}} \) and \( 1 - \lambda_{w_{i-n+1}...w_{i-1}} \) are estimated using a development set, and observe the following properties:

\[
\sum_n \lambda_n = 1
\]

\[
\bigwedge_n 0 \leq \lambda_n \leq 1
\]

In spite of their conceptual shortcomings, smoothed \( n \)-gram language models are still widely used in natural language processing, as they are easy to train, fast in decoding, and require no manual annotation. These properties allow the use of very large amounts of training corpora to be employed, which helps to compensate for the limited, surface-oriented history.

There’s no doubt that \( n \)-gram models have been successful in machine translation, speech recognition, and other areas for the past few decades, and that their use will continue for years to come. But they do have their limits. Researchers at Google (Brants et al., 2007) explored using very large \( n \)-gram language models in the context of machine translation. They trained such models on corpora ranging from 13 million English tokens to 2 trillion tokens. They found increases in \textsc{Bleu} score (cf. §5.3.3) on the NIST OpenMT 2006 Arabic-English test set, when trained on up to around 100 billion tokens, shown in Figure 3.1 (Brants et al., 2007, p. 865).

They state that a linear least-squares regression of their logarithmic improvement has \( R^2 = 0.96 \). They thus infer that their \textsc{Bleu} score improvements have logarithmic growth relative to the number of tokens in the training set. Their growth is in fact better modeled by iterated logarithmic growth (cf. Cormen et al., 2001)—a much
Figure 3.1: BLEU scores on NIST OpenMT 2006 Arabic-English test set by Brants et al. (2007), using either interpolated Kneser Ney smoothing (Chen and Goodman, 1998) or their unnormalized Stupid Backoff.
slower growth function:

\[
\log^* n \triangleq \min \{ i \geq 0 : \log^{(i)} n \leq 1 \}
\]  

(3.5)

A log-logarithmic growth model (log ◦ log) gave \( R^2 = 0.9762 \), and an approximated iterated logarithmic growth model gave \( R^2 = 0.9839 \). Thus the iterated logarithmic growth model better accounts for their results.

These findings are important because they show the diminishing gains from simply training an \( n \)-gram language model on more and more data. \textit{Brants et al. (2007)} obtained less than a half of a BLEU point increase by adding 900 billion tokens. Adding another \textit{quadrillion} English tokens would give them a BLEU score of \( \approx 49.86 \), by their own flawed logarithmic growth model. The much higher-correlated iterated log growth model would give \( \approx 47.24 \). These figures are summarized below:

<table>
<thead>
<tr>
<th>Growth</th>
<th>( R^2 )</th>
<th>BLEU @ quadrillion tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>log</td>
<td>0.96</td>
<td>49.86</td>
</tr>
<tr>
<td>log log</td>
<td>0.976</td>
<td>47.92</td>
</tr>
<tr>
<td>( \log^* )</td>
<td>0.984</td>
<td>47.24</td>
</tr>
</tbody>
</table>

...3In order for their system to get a BLEU score of 65, Google would need to accumulate one googol English tokens \((1 \times 10^{100} = \text{ten duotrigintillion})\). Pun intended.
domains. Adding all of the text in the English-speaking Internet into an $n$-gram language model will not solve the problem.

Alternative approaches to language modeling have been formulated since the advent of $n$-gram language models, and these are described below.

### 3.1 Cache Language Models

The production of language is not a truly ergodic process. The same words often occur again and again in a given document, with much greater frequency than a general corpus. For example, the word *language* occurs in this document much more frequently than in most English-language documents. One approach to account for this is to separate documents in a corpus by genre, such as politics, sports, entertainment, etc. Then we could build different language models for each genre, and tune genre weights on held-out data. However, we would need to know beforehand what genres exist in a given corpus, and we would need to have held-out texts that are representative of the test set. Furthermore, subgenres exist in a corpus, so that the terminology and names for discussing, say, basketball, are quite different from those used in discussing hurling.

Kuhn and De Mori (1990) built on the idea that words tend to occur in bursts by developing a *cache* language model. This model was not intended to function independently, but rather is used in conjunction with what is now usually referred to as a class-based model (cf. Derouault and Mérialdo, 1984, 1986; Brown *et al.*, 1992a). The relative weighting of their cache model and class model depends on the
part-of-speech of the word being predicted. Each part-of-speech has different weights for the cache: class ratio.

The cache model itself consists of a buffer of 200 words (by default) for each part-of-speech. Word types may only have one part-of-speech, which in that work are manually-created classes, rather than inferred classes like those in Brown et al. (1992b) or Kneser and Ney (1993). As soon as a given buffer has at least five words in it, it begins producing a maximum likelihood probability distribution. Newer words replace the least used words in the buffer. The idea for the cache language model was inspired by the use of memory caches in computer hardware designs, which locally store the contents of recently-used memory addresses.

Cache language models tend to perform well on small- and medium-sized textual experiments, but they do not perform as well on speech recognition tests, since missed predictions are compounded in the cache (Goodman, 2000, 2001, pp. 3 & 3,25 resp.), although using a trigram cache can help (Jelinek et al., 1991).

3.2 Skip Language Models

Huang et al. (1993) took the basic idea behind n-gram models and allowed for intervening words between the predicted word and a given word upon which the predicted word is conditioned. That is, in addition to using a basic bigram model \( P(w_i|w_{i-1}) \), a skip language model\(^4\) would also use \( P(w_i|w_{i-2}) \), as well as \( P(w_i|w_{i-3}) \), and so forth up to an arbitrary distance \( k \). This model could also generalize to

\(^4\)These were originally called long-distance n-gram language models.
n-grams in a straightforward manner, using for example $P(w_i|w_{i-3} w_{i-2})$ and even more exotic configurations like $P(w_i|w_{i-6} \ldots w_{i-3})$ and $P(w_i|w_{i-6} w_{i-4} w_{i-2})$. Guthrie et al. (2006) define the set of $k$-skip-$n$-grams in a sentence $w_1 \ldots w_m$ as:

$$S_k = \{w_i, w_{i+1}, \ldots w_i \mid \sum_{j=1}^{n} i_j - i_{j-1} < k\}$$

These all are interpolated in the usual manner, such that the weights $\lambda$ observe the following properties:

$$\sum_{n} \lambda_n = 1$$

$$\bigwedge_{n} 0 \leq \lambda_n \leq 1$$

These weights are optimized on held-out data. As would be expected, increased distance between a predicted word and its conditioning word(s) results in increased perplexity (Goodman, 2001, §4). Skip language models have seen moderate success in various experiments (cf. Goodman, 2000, 2001; Guthrie et al., 2006; Brun et al., 2007; Chong et al., 2013; Pickhardt et al., 2014).

Ronald Rosenfeld, the creator of skip language models, described them as “seriously deficient”, saying “Although they capture word-sequence correlations even when the sequences are separated by distance $d$, they fail to appropriately merge training instances that are based on different values of $d$. Thus they unnecessarily fragment the training data.” (Rosenfeld, 1994, pg. 16).
3.3 Trigger Language Models

Cache language models (§3.1) use the idea that if a word occurs once in running text, it will likely occur again—maybe in the same sentence, or in subsequent sentences. But intuitively we know that not only will that word likely occur again, but associated words as well. For example, if we have seen the word doctor, then we might also reasonably expect to see the word nurse, patient, or treatment soon thereafter. But how do we know that doctor is associated with patient? We could look at how often the two words occur in the same sentence or document of a corpus.

Ronald Rosenfeld and colleagues developed this intuition into trigger language models (Rosenfeld and Huang, 1992; Lau et al., 1993b,a; Rosenfeld, 1994, 1996). They used average mutual information (cf. Abramson, 1963) as the core measurement of whether word(s) \( A_h \) in the history triggers the next word(s) \( B \):

\[
I(A_h : B) = P(A_h, B) \log \frac{P(B|A_h)}{P(B)} + P(A_h, \overline{B}) \log \frac{P(\overline{B}|A_h)}{P(B)}
\]

\[
\quad + P(\overline{A_h}, B) \log \frac{P(B|\overline{A_h})}{P(B)} + P(\overline{A_h}, \overline{B}) \log \frac{P(\overline{B}|\overline{A_h})}{P(B)}
\]

They used this measurement to ensure that more frequent co-occurrences receive a higher score, while also discounting words that occur frequently in their own right.
The value of average mutual information is symmetric and non-negative. When the log function is in base 2, the value $I(A_h : B)$ corresponds to the average number of bits saved by predicting $B$ given $A_h$, rather than predicting $B$ without being given $A_h$ (Rosenfeld and Huang, 1992).

Recording the average mutual information between pairs of words (i.e. one word triggering another, possibly non-adjacent word) requires a very large number of parameters, much closer to $V^2$ than bigram language models because adjacency is not a requirement. Therefore for every word $B \in V$ Rosenfeld (1996) keeps only the top $k$ triggers having the highest average mutual information associated with $B$, where $k$ is either three or six in their experiments. They found that quite often a word triggers itself—such self-triggers were found in the top 6 for 90% of all the words in the vocabulary. Thus the main idea in cache language models forms an important part of trigger language models as well.

Rosenfeld and Huang (1992) tried different methods of combining multiple triggers from the history, either by using the most informative trigger, using the arithmetic mean of all the triggers, or using the sum thereof. They found no significant difference between these three methods. They then linearly interpolate the trigger model with a trigram model, finding a 10% improvement over a simple trigram baseline on Wall Street Journal (WSJ) data, and larger gains for Brown corpus (Kučera and Francis, 1967) test data. Rosenfeld (1996) incorporated triggers into a maximum entropy model (see Section 3.6).
3.4 Topic-based Language Models

Gildea and Hofmann (1999) model language as being generated by topics, which in turn are generated by previous words. This topic-based model is illustrated graphically in plate notation in Figure 3.2. An example sentence showing topics generating words, which in turn generate subsequent topics, is shown in Figure 3.3. Intuitively we could think of different topics such as politics or sports as generating different word distributions. Two questions naturally arise here: first, how do we determine what topics exist; and second, how do we know what topic a sentence or document belongs to?

The first question of determining what topics exist could be made by news editors or corpus annotators. Alternatively, we could try to induce a predetermined number of topics in an unsupervised manner. Gildea and Hofmann (1999) take this latter approach. The latent topics are ultimately derived from counts of words in documents, reminiscent of latent semantic analysis (Deerwester et al., 1990), but using a probabilistic interpretation with expectation maximization (Hofmann, 1999). First, they count the frequency of words in different documents, storing these in a term–document matrix \( N \). Then they use the expectation maximization (EM) algorithm (Dempster
et al., 1977) of alternating between calculating the probability that a given word $w$ in a given document $d$ was generated by a topic $t$, and adjusting the model parameters based on the previous step. These induced topics may not resemble any document classification scheme proposed by humans.

They answer the second question, how do we know what topic a sentence or document belongs to, in a similar manner. The basic idea is to use all the words seen so far in a document to predict the topic. They use the same basic technique of EM described above, using only one iteration per predicted word to minimize computational load.

Now that we can determine what topics exist, and what topic(s) a given document belong to, we can use the topic(s) to generate words. In other words, we use the idea of topics as a bottleneck variable between the myriad words in our sentence/document.
history, and the predicted word. This notion is expressed as:

$$P_t(w|h) \triangleq \sum_t P(w|t)P(t|h)$$  \hspace{1cm} (3.8)

where \(w\) is a word, \(t\) is a topic, and \(h\) is a word history. We can see that this is a bag-of-words model—the order of the words in the history is ignored. They only predict a topic, which in turn predicts a word. This is one of the main shortcomings of topic-based language models. The previous word or two greatly contribute to the prediction of the subsequent word (Goodman, 2001, pg. 14), which is why bigram and trigram language models perform so well on their own. The authors tackled this issue by interpolating their model with a trigram model.

In their experiments they used a simple trigram model (no smoothing method specified) as their baseline, and they do not include the end-of-sentence marker (</s>) in the calculation of perplexity. This latter omission leads to a higher improvement over an already weak trigram model baseline due to the bag-of-words nature of their model. On a news corpus of 6.8M words, with linear interpolation they saw a 7.8% improvement over the basic trigram model, and they saw larger gains for log-scale interpolation (Klakow, 1998) and unigram rescaling. Unigram rescaling was defined as:

$$P_{ur} \propto P_{tri} \frac{P_{topic}}{P_{unigram}}$$  \hspace{1cm} (3.9)

and log-scale interpolation as:

$$P_{ls} \propto P_{tri}^\lambda P_{topic}^{1-\lambda}$$  \hspace{1cm} (3.10)
Both log-scale interpolation and unigram rescaling require costly normalization, with each queried sentence having a time complexity of $O(|w| \times |V|)$, where $w$ is a sentence and $V$ is the vocabulary. Larger-scale experiments using a trigram model trained on 450M words also saw similar improvements using a topic-based model, but these did not translate to improvements in speech recognition experiments. The topic+trigram model using unigram rescaling gave a worse word error rate than a basic trigram model.

Subsequent works have focused on different interpolation methods, as well as adapting the training method to work on computer clusters (cf. Tan et al., 2011, 2012).

3.5 Neural Network Language Models

3.5.1 Feedforward Neural Networks

The idea of using an artificial neural network\(^5\) (Rosenblatt, 1957; Gamba et al., 1961; Widrow, 1962) for natural language processing has been around for quite some time. Miikkulainen and Dyer (1991) used a single hidden-layer feedforward neural net with standard backpropagation (Werbos, 1974; Parker, 1985) to learn semantic role labels from incremental language input, as well as to generate paraphrases. The network worked to reduce many dimensions of variation into a lower dimension representation. This basic idea was also employed in latent semantic analysis (Deerwester et al., 1990).

\(^5\)For introductions to artificial neural networks, see Russell and Norvig (2003, §20.5) and Wikiversity.org.
Xu and Rudnicky (2000) implemented a bigram neural network language model for use in speech recognition. They used a fully connected single layer network (i.e. no hidden layers) with no bias weight. Their input layer and their output layer each had \(|V|\) nodes, with one-hot encoding. They trained their network using the standard backpropagation algorithm (Werbos, 1974; Parker, 1985), with initial weights set to 0, and they used batch updating. Due to the expansive size of the network and high computational cost, they evaluated their model on a very small corpus with a vocabulary of about 2500 words. They achieved comparable word error rates as Kneser-Ney models (Kneser and Ney, 1995), although training took several orders of magnitude longer. Each training epoch took about twice as long as training an entire Kneser-Ney model, and their model required thousands of epochs. They noted an interesting future direction for their work, however: “adding recurrent connections so that the network can model the long distance dependencies.” The realization of this suggestion is the primary contribution of Mikolov et al. (2010), which we discuss later.

Like Miikkulainen and Dyer (1991), Bengio et al. (2000) used the idea of a single hidden-layer feedforward neural network to reduce high-dimensional discrete distributions for natural language (see also Bengio and Bengio, 2000a,b). Their architecture is represented in Figure 3.4. Like previous works, word types are represented as a real vector of fixed length \(m\), where \(m\) represents the number of dimensions in a word feature space \(\mathbb{R}^m\). The mapping \(C\) of a word type \(i \in V\) is realized as a \(|V| \times m\) matrix, where \(V\) is the vocabulary—the set of words in a language.

The primary contributions of Bengio et al. (2000, 2003, pp. 5 & 1140 resp.) were the large-scale application of such a network, and learning probabilistic models of
word sequences. They used a hyperbolic tangent hidden layer and a softmax output layer to ensure a probability distribution. Thus most of the computation occurs in the normalization of the hidden layer’s output. For this they parallelized the normalization for the output layer. Their first attempt was to employ an asynchronous shared-memory [sic] architecture, which inevitably leads to race conditions. They reasoned that this occurred infrequent enough so as to not be a major problem. Their second attempt was to employ an asynchronous message-passing architecture, which duplicated onto different nodes in a network all the data necessary to calculate the normalization for the output layer.

**Bengio et al. (2003)** evaluated their model on two corpora: the Brown corpus (Kučera and Francis, 1967) and an AP newswire corpus. The Brown corpus is a...
balanced corpus of 1.2M English tokens, 800K of which they used for their training set. The AP news corpus consisted of 16M English tokens from 1995–1996 articles, 14M of which were used for their training set. Training on the Brown corpus converged after 10–20 epochs, while they had to cut short training for the AP corpus. They ran five epochs on that corpus, which took three weeks using 40 CPUs. They compared their model with backoff $n$-gram models, even though it was well-known by then that the interpolated Kneser-Ney counterpart perform better (Chen and Goodman, 1998, pg. 39), and that Modified Kneser-Ney performs better than Kneser-Ney (Chen and Goodman, 1998, pg. 40). They also used an unusual manner of reporting perplexity improvements, stating the percentage that the (backoff) baseline is higher relative to the new model:

$$\Delta_{PP} \triangleq \frac{PP_0}{PP_1} - 1$$

(3.11)

rather than stating the percentage that the new model reduces perplexity relative to the baseline:

$$\Delta_{PP} \triangleq 1 - \frac{PP_1}{PP_0}$$

(3.12)

This has the effect of inflating percentages.

Here, we report the reduction in perplexity relative to backoff 4-gram Kneser-Ney (i.e. using Equation 3.12). Table 3.1 shows their results on the Brown corpus and the AP news corpus. The results for the small Brown corpus are quite impressive, while those for the AP news corpus are good. For the Brown corpus, they used a history of 4, with 100 hidden nodes, 30 word features, and they interpolated with a trigram model, which reduced perplexity over a backoff 4-gram Kneser-Ney model for
Table 3.1: Experimental perplexity results of Bengio et al. (2003)’s feedforward neural network with one hidden layer, trained on either 800K words of the Brown corpus or 14M words of an AP news corpus.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>4KN-Backoff PP</th>
<th>+FF-NNet PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN Brown 800K</td>
<td>321</td>
<td>252</td>
<td>21.5%</td>
</tr>
<tr>
<td>EN AP 14M</td>
<td>119</td>
<td>109</td>
<td>8.4%</td>
</tr>
</tbody>
</table>

the 800K word Brown corpus by 21.5%. For the AP news corpus, they used a history of 5, with 60 hidden nodes, 100 word features, and they interpolated with a trigram model, which reduced perplexity over a backoff 4-gram Kneser-Ney model for the 14M word AP news corpus by 8.4%.

Subsequent works on these models have shown their utility in speech recognition (eg. Schwenk and Gauvain, 2002; Schwenk, 2007) and machine translation (eg. Schwenk, 2010; Schwenk et al., 2012).

### 3.5.2 Recurrent Neural Networks

Feedforward networks do not have cycles, which simplifies learning. However, temporal information is dealt with quite indirectly in these models. Jordan (1986) and Elman (1990) allowed for cycles as a way of more explicitly encoding temporal information. The basic structure of Elman networks\(^6\) has an input layer, hidden layer(s), and an output layer, just like feedforward neural networks do. However, they also have another hidden layer, called the context layer or state. The normal hidden layer

\(^6\)Also called simple recurrent networks.
effectively copies itself to the context layer by means of feeding into the context layer with weights set to 1. Then the context layer feeds into the original hidden layer at the next time step. This means that at time step \( t \) the hidden layer effectively receives inputs from the input layer as well as from its own layer at time \( t - 1 \). This structure is illustrated in Figure 3.5 from Elman (1990).

In order to learn optimal weights, a modification of the standard backpropagation algorithm (Werbos, 1974; Parker, 1985) is employed. The most common technique is called backpropagation through time (BPTT) (Rumelhart et al., 1986; Werbos, 1988, 1990) This algorithm essentially unfolds a recurrent network through time, then operates like standard backpropagation. This is possible because recurrent networks are equivalent to (a more complex) feedforward network, assuming a finite length of time \( \tau \) (Rumelhart et al., 1985), the same way that any formal grammar is equivalent to its immediately less-expressive grammar assuming an unbounded part of the grammar becomes bounded.\(^7\) This is illustrated in Figure 3.6, from Bodén (2002).

\(^7\)For example, indexed grammars allow all child nonterminals to receive copies of the stack of their parent node, while linear indexed grammars only allows for one child nonterminal to receive a copy of its
Figure 3.6: Left: Elman network (Elman, 1990); Right: unfolded Elman network with $\tau = 3$. From Bodén (2002).
Mikolov et al. (2010) used an Elman network (Elman, 1990) to model language. The number of hidden units ranged from 30 to 500. For training they used truncated backpropagation through time, which is simply BPTT with a history ($\tau$) of one. The learning rate, $\alpha$, started at 0.1, and if at the end of each epoch the development set log-likelihood did not significantly increase, the learning rate was halved; otherwise $\alpha$ stayed the same for the next epoch.

They compared the number of parameters that need to be selected ad hoc before training in an Elman network with a feedforward network, and state that Elman networks only have one such parameter (size of hidden layer) compared with three for feedforward networks (size of hidden layer, history length, and size of projection layer). However, this is not entirely true: they have chosen to set their training history length $\tau = 1$ in backpropagation through time. This $\tau$ functions somewhat differently than the history length parameter in feedforward networks, but it is a parameter that needs to be selected ad hoc before training nonetheless. Another ad hoc pre-training parameter in their system, not specific to their network configuration, was a threshold for merging infrequent words to a special rare token.

Like Bengio et al. (2003), Mikolov et al. (2010) encountered great difficulty training all but small-scale models. They note: “[The] time needed to train optimal network increases faster than just linearly with increased amount of training data: vocabulary growth increases the input and output layer sizes, and also the optimal hidden layer size increases with more training data” (Mikolov et al., 2010, pg. 2). This superlinear parent’s stack. Similarly, we could unroll a regular grammar to become a finite grammar by replacing the rule “$A \rightarrow bA$” with a finite depth of recursion, such as $A \rightarrow bA_1; A_1 \rightarrow bA_2; A_2 \rightarrow bA_3; A_3 \rightarrow b$.
growth has been the major impediment to the adoption of neural network language models, and most of the research in recent years has involved tackling this issue.

While their training set consisted of 37M words from the New York Times section of the English Gigaword corpus, they were unable to train it, so they used 6.3M words (300K sentences) of it. The best-performing models, having a larger hidden layer, took several weeks to train.

In their experiments they unfortunately compared their model with backoff versions of modified Kneser-Ney n-gram models, even though it was well-known by then that the interpolated modified Kneser-Ney counterpart performs better (Chen and Goodman, 1998, pg. 39, 40). Their first experiment involved what is referred to as the DARPA WSJ’92 and WSJ’93 datasets, presumably Garofalo et al. (1993, 1994). This involves a 100-best list with an oracle word error rate of 9.5%, with training data coming from the above-mentioned New York Times section of an unspecified version of the English Gigaword corpus (cf. Graff and Cieri, 2003, *inter alia*). They saw a 9.9% reduction in word error rate over a 5-gram Kneser-Ney backoff model when they combined three of their RNN models with a 5-gram Kneser-Ney backoff model. The perplexity reductions on the development set also look very good, although no perplexity results are reported on the test set.

They note that the acoustic models for this dataset are very weak by modern standards, so they evaluate their model on a newer dataset. They reranked the lattices from the Augmented Multiparty Interaction (AMI) system on the NIST

---

8No version information was specified.
9http://corpus.amiproject.org
RT05 evaluation (Hain et al., 2006). Their baseline language model was a 4-gram *Jelinek-Mercer* model (Hain et al., 2006, § 4.4) trained on 1.34B words of various sources, which had a word error rate of 24.1%. A 500-node RNN having a frequency cutoff of 10, interpolated with the 4-gram Jelinek-Mercer model had a word error rate of 23.3%, a reduction of 3.3%.

Since this work appeared most of the effort in this area has gone to improving training times and scalability (Mikolov et al., 2011; Mnih and Teh, 2012, *inter alia*), by means of training an RNN in conjunction with a maximum entropy language model (Rosenfeld, 1994), as well as reducing the number of training epochs, reducing the training set size, reducing the vocabulary, reducing the size of the hidden layer, parallelizing training, and using SIMD instructions where possible. The combination of these techniques has allowed RNN language models to be trained on larger training sets. Google researchers trained an 800-node RNN on 825M tokens in two weeks, using 16 CPUs (Chelba et al., 2014).

### 3.6 Maximum Entropy Language Models

The question in trigger language models (§ 3.3) of how to incorporate multiple triggers evolved into a more general one in Lau *et al.* (1993b) of how to reliably incorporate multiple sources of knowledge.

Backoff language models (Katz, 1987) use the most reliable source when sufficient data exists from such source. For example, given counts of trigrams, bigrams, and unigrams from a training corpus, a backoff model will give a probability distribution
for the next word if there are enough occurrences of the previous two words in the training set. If not, then it will “backoff” to the lower order, using bigram counts, and so forth. This is expressed in the general form (Chen and Goodman, 1998) as:

\[
P_{BO}(w_i|w_{i-n+1} \ldots w_{i-2}) = \begin{cases} 
\alpha(w_i|w_{i-n+1} \ldots w_{i-1}) & \text{if } c(w_{i-n+1} \ldots w_i) > 0 \\
\gamma(w_{i-n+1} \ldots w_{i-1})P_{BO}(w_i|w_{i-n+2} \ldots w_{i-1}) & \text{if } c(w_{i-n+1} \ldots w_i) = 0 
\end{cases}
\]

Linearly interpolated language models (Jelinek and Mercer, 1980) take a different approach, using all sources to varying degrees. Non-negative weights are assigned to each source (eg. trigram, bigram, unigram) such that all weights sum to one. The linearly interpolated distribution is then the sum of the probability from each source \(s\) times its associated weight. This is expressed in a general form as:

\[
P_{JM}(w|h) \triangleq \sum_{s=1}^{|s|} \lambda_s P_s(w|h) \tag{3.13}
\]

Lau et al. (1993b) use the principle of maximum entropy (Jaynes, 1957) to develop an alternative method of incorporating multiple sources of information. In this method, myriad sources of information are formulated into a single model as various constraints. There may be many possible probability functions that satisfy all given constraints. If there exists more than one, then choose the probability function that has the highest entropy, so that nothing else is assumed.

The authors use triggers as a set of constraints, and search for the highest entropy probability function using Generalized Iterative Scaling (GIS: Darroch and Ratcliff,
GIS is an optimization algorithm designed for maximum entropy modeling, and has largely been superseded by L-BFGS (Liu and Nocedal, 1989) for large scale maximum entropy models due to the latter’s faster convergence. Lau et al. (1993b) trained 5M words of news text (WSJ), which took 500 MIPS-hours per iteration, and incorporated a trigram model into their maximum entropy model. They saw a 12% reduction in perplexity over a basic trigram model. A concurrent paper by the same authors (Lau et al., 1993a) report a 23–27% improvement over a basic trigram model, using a closed vocabulary of 20K words and more trigram constraints.

Chen (2009, *inter alia*) built on this general framework to develop Model M, a class-based maximum entropy model using the shrinking criterion described in Chen et al. (2009). This criterion seeks to reduce an \( n \)-gram maximum entropy model by looking for similarities among regularized parameter estimates. When these are found, a new feature is added that is the sum of the original features. The word classes that they use come from the agglomerative clustering algorithm described in Brown et al. (1992a).

They evaluated using the same basic setup as Lau et al. (1993b) (Paul and Baker, 1992), but only trained on up to 900K words. For the model trained on 900K words, they saw an 11.1–13.3% reduction in perplexity over a 4-gram modified Kneser-Ney baseline (Chen and Goodman, 1998), depending on the number of word classes ranging from 50–500. Word error rates went from 22.3% to 20.8% (a 6.7% improvement), using the original (weak) acoustic models.
Table 3.2: Comparison of different language models. Distance: 1=short: no intervening words between history and predicted word; 2=medium: distance-dependent history with intervening words; 3=long: arbitrary dependencies/relations within sentence, 4=long across sentence boundaries.

<table>
<thead>
<tr>
<th>Language Model</th>
<th>Incremental</th>
<th>Lexical</th>
<th>Distance</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-gram</td>
<td>Y</td>
<td>Y</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>Class</td>
<td>Y</td>
<td>N</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>Cache</td>
<td>Y</td>
<td>Y</td>
<td>4</td>
<td>N</td>
</tr>
<tr>
<td>Skip</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>N</td>
</tr>
<tr>
<td>PCFG</td>
<td>N</td>
<td>N</td>
<td>3</td>
<td>N</td>
</tr>
<tr>
<td>PLSA</td>
<td>Y</td>
<td>N</td>
<td>3</td>
<td>Y</td>
</tr>
<tr>
<td>Trigger MaxEnt</td>
<td>Y</td>
<td>Y</td>
<td>4</td>
<td>Y</td>
</tr>
<tr>
<td>Class MaxEnt</td>
<td>Y</td>
<td>N</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td>FF-NN</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td>RNN</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>Y</td>
</tr>
</tbody>
</table>

3.7 Summary

"Thou whoreson Zed! thou unnecessary letter!"

–William Shakespeare, King Lear

This chapter has given a broad overview of some notable language models, especially those commonly used in speech recognition settings. Some attributes of these models are presented in Table 3.2, with the column Z referring to whether or not costly normalization over the entire vocabulary is required to produce a probability. Such normalized models are generally discriminative.
Other types of language models, such as probabilistic context-free grammars (Baker, 1979), structured language models (Chelba and Jelinek, 1998; Chelba, 2000), whole-sentence language models (Rosenfeld et al., 2001), and other syntax-oriented language models are not used as often in speech recognition systems as they are in machine translation systems and natural language generation systems. There is a theoretical justification for this.

The tasks of speech recognition, optical character recognition, and software keyboards do not involve permuting output signals \( x_t \) from one time step \( t \) to the next. Rather, input signals \( y_t \) are effectively substituted to output \( x_t \), possibly subject to interactions between time steps (e.g., coarticulation). Language models are used here to ensure \textit{lexical coherency} and to a lesser extent \textit{local fluency}. For example, given an audio signal at time \( t = 6 \), an acoustic model might propose the top three candidates are \textit{ban}, \textit{pan}, and \textit{pant}. If we have hypothesized that the previous words are \textit{cooked the fish in the}, then a language model having sufficient history can promote \textit{pan} here.

At the other end of the spectrum, natural language generation systems rely on language models to ensure overall grammatical fluency to a much greater degree. Input signals, as a sentence-level logical form, are subject to broad structural changes as they are realized in a given language, such as English or Japanese. Lexemes must indeed be realized among synonym candidates, but their choice is not nearly as detrimental to realization as overall grammatical fluency. This is where unlexicalized and/or syntactic language models can most contribute. Machine translation systems make use of language models both for lexical coherency and for grammatical fluency, although more so for fluency.
Chapter 4

Description of the Neurophysiologically-inspired Language Model

4.1 Overview

This model incorporates an ELAN-based component, an N400-based component, a P600-based component, and a component that uses local lexical history. We use these various components in tandem to overcome the shortcomings of each of these on their own. We first provide a broad overview of the model, then discuss each of these components of the statistical language model in their own subsections below.

The probability of a sentence in our model is defined as:

\[ P_{\nu\phi}(w_1 \ldots w_k) = \prod_{i=1}^{k} P(C_i|C_{i-m} \ldots C_{i-1})P(w_i|C_i, w_0 \ldots w_{i-1}) \]

where \( C_i \) is the class of the word at the \( i^{th} \) position, \( k \) is the length of the sentence, and \( w_0 \) is the start symbol.

The emission probability is a linear interpolation (cf. Jelinek and Mercer, 1980; Kneser and Steinbiss, 1993; Klakow, 1998) of class emissions, word \( n \)-gram conditional probabilities, and a full-sentence-history lexical probability:

40
\[ P_{\nu\phi}(w_i|C_i, w_0 \ldots w_{i-1}) \triangleq \lambda_C P_C(C_i|C_{i-m+1} \ldots C_{i-1})P_C(w_i|C_i) \\
+ \lambda_{E} P_{E}(w_i|w_1 \ldots w_{i-1}) \\
+ \lambda_{n} P_{n}(w_i|w_{i-n+1} \ldots w_{i-1}) \]

Linear interpolation is used because we have a small number of features (cf. Bengio, 2002, §2.5), and it allows us to preserve a (generative) probabilistic model without needing to resort to expensive calculation of the normalization term \(Z\). Each component here is fast, probabilistic, incremental, and complementary.

We now discuss each of these components in more detail.

### 4.2 ELAN-based Language Model Component

As discussed in section 2.1, the ELAN is involved in local, first-pass decoding of word categories. We employ a statistical language model counterpart to the ELAN that exhibits similar properties. It is local, having a limited history, rather than being able to make use of the full sentence history. Similarly it is first-pass or greedy, taking the preferred search path. It focuses on word categories, rather than either visible words or phrasal categories. We describe in greater detail below a statistical language model component that exhibits each of these properties.

We can model the probability of a words \(w_1 \ldots w_k\) based on the probability of a class \(C_i\) emitting word \(w_i\) times the probability of \(C_i\) following \(C_{i-1}\). Equation 4.1 describes a first-order hidden Markov model, illustrated afterwards as a dynamic
Bayesian network:

\[ P_{HMM}(w_1 \ldots w_k) \triangleq \prod_{i=1}^{k} P(C_i|C_{i-1})P(w_i|C_i) \]  \hspace{1cm} (4.1)

Second-order hidden Markov models increase the history used in estimating \( P(C_i|h) \) to the previous two states:

\[ P_{HMM2}(w_1 \ldots w_k) \triangleq \prod_{i=1}^{k} P(C_i|C_{i-2}, C_{i-1})P(w_i|C_i) \]  \hspace{1cm} (4.2)

As Formulas 4.1 and 4.2 show, word categories are conditioned on the previous \( m \) states, where \( m = 1 \) for first-order models, \( m = 2 \) for second-order models, and so forth. Thus this model uses a limited, local history, rather than making use of the entire sentence history. This stands in contrast to models like Jelinek and Lafferty.
which ultimately use the entire sentence history in predicting prefix probabilities.

The class-based language model of Brown et al. (1992a) is based on a second-order hidden Markov model. Rather than relying on a supervised part-of-speech tagger trained on a manually-annotated corpus, they induced word classes from unannotated text, using a greedy algorithm that indirectly attempts to maximize the average mutual information between adjacent classes, which is defined as:

\[
I(C_1, C_2) \triangleq \sum_{C_1 \in C} \sum_{C_2 \in C} p(C_1, C_2) \log \left( \frac{p(C_1, C_2)}{p(C_1)p(C_2)} \right)
\]

(4.3)

They first initialize each word type as its own word class, successively merge pairs of classes that result in the least loss of average mutual information, stopping once the predefined number of classes is reached. They then iteratively swap words to the class resulting in the greatest average mutual information, until convergence. Since the classification of words is deterministic, the class states are not hidden, allowing easy inference. We must still model \( P(w|C) \) to ensure a probability distribution.

Brown et al. (1992a)’s class-based model reflect the unlexicalized, localized properties of the ELAN event-related potential. This is expressed as:

\[
P_C(w_1 \ldots w_k) \triangleq \prod_{i=1}^{k} P(C_i|C_{i-m+1} \ldots C_{i-1})P(w_i|C_i)
\]

(4.4)

where \( C_i \) is considered here as a function for producing word classes given a surface-form
word $w_i$, which is trained using the exchange algorithm with orthotactic information (Kneser and Ney, 1993; Martin et al., 1998; Clark, 2003). In contrast to the agglomerative approach of the Brown algorithm described above, the exchange algorithm initializes the $G$ most frequent word types as their own respective classes, the rest of the word types falling into a single class. It then iteratively moves words from one class to the class that results in the lowest perplexity, until convergence. Clark (2003) adds a bias that favors assigning words to classes that are orthographically similar.

The ease at which we can estimate a class-based language model permits us to have a longer history with more reliability, interpolating with lower-order histories or backing-off when necessary.

4.3 N400-based Language Model Component

“To decompose is to live too”
–Samuel Beckett

The ELAN-based language model component described above abstracts beyond words, so that we can model the probability of, say, a noun following a determiner. But this comes at the expense of modeling specific lexical relationships. That is, the observed variables are not (directly) conditioned on previously-observed variables. This can have a large impact on language models with respect to idiomatic expressions
and lexical priming, for example. Consider below an example using an HMM with Penn Treebank tags for perspicuity:

The next word will be conditioned on the next tag, which itself is conditioned on the previous \( n - 1 \) tags. It’s easy to see that this model will not allocate, comparatively, enough probability to what we would be expecting to follow, namely \textit{dogs}. Using data from the Penn Treebank, the most likely next word will be \textit{shares}. Thus we would like to integrate lexical information from the sentence’s history. And Example (1) has shown that local lexical history (as in \textit{n}-gram models) may also not be enough in certain cases.

The N400 can help inform us as to how lexical-semantic relations are processed neurophysiologically. Various characteristics of the N400 described in section 2.2 include the following:

1. The N400 involves associations between content words
2. The context for the N400 is cumulative
3. Longer context helps constrain negativity
4. The frequency of a target word affects the magnitude of negativity
5. The N400 is not sensitive to syntactic constraints
6. Closed-class words in the context do not facilitate processing of open-class target words
These characteristics have implications in designing an N400-based statistical language model component. Let us look at each of these in detail.

Characteristic 1: Involves associations between content words. Firstly, this compels us to both distinguish content words from function words. We implement this by using the multiplicative inverse of the count of the context word in a balanced training set. We also need to measure the lexical associations between words of the former group, such as using the log-likelihood ratio (Dunning, 1993).

Characteristic 2: The context for the N400 is cumulative. Thus we should use the full sentence history, or at least a much longer history than what is typically employed in n-gram language models. As a result of this property and the previous property, we must abandon the joint, contiguous nature of histories as they are used in n-gram models. Rather, histories in our model are decomposed in various way to best suit the component. For the N400 component, this means each individual word in \( w_1 \ldots w_{i-1} \) contributes to the prediction of \( w_i \).

Characteristic 3: Longer context helps constrain negativity. As the sentence progresses from the start towards the end, the addition of words in \( w_1 \ldots w_{i-1} \), particularly content words, gives rise to a less uniform distribution of \( P_t(w_i|h) \). This further makes the case for setting \( h \triangleq w_1 \ldots w_{i-1} \); otherwise the Markovian assumption would prevent the observed constraint in negativity.

Characteristic 4: The frequency of a target word affects the magnitude of negativity. \textit{Van Petten and Kutas} (1990) and subsequent work found an inverse relationship between frequency of content words and N400 amplitude. We implement this by using the multiplicative inverse of the count of the predicted word, as estimated from a
balanced training set.

Characteristic 5: The N400 is not sensitive to syntactic constraints. Kutas and Hillyard (1983, p. 543, 546), among many others, have shown that grammatical violations do not elicit an N400. Rather, this event-related potential is sensitive to lexical semantic content. Thus our N400-based component does not account for structural well-formedness, but treats \( h \) as a multiset subject to soft memory constraints (cf. Van Petten and Kutas, 1991).

Characteristic 6: Closed-class words in the context do not facilitate processing of open-class target words. We implement this by weighting closed-class words in \( h \) low, minimizing their contribution to the prediction of the target word.

As such, we define the initial N400-based language model component as:

\[
P_{\ell_0}(w_i|w_1 \ldots w_{i-1}) \triangleq \frac{Z(h)}{Z}\sum_{j=1}^{i-1} a(w_j, w_i) e^{-\lambda_d(i-j)} c'(w_j)^p_h p_i (4.5)
\]

Where \( a \) is an association measure, such as the log-likelihood ratio or average mutual information; \( Z(h) \) is the partition function, so that \( \sum_w p(w|h) = 1 ; \lambda_d \) is a decay constant; \( p_h \) is a history frequency penalty which controls the degree to which rarer words in the history generate \( w_i \); and \( p_i \) controls the degree to which rarer words are generated. Here \( c'(\cdot) \) represents a smoothed count to allow for unseen words, using Good-Turing estimation.

Both scaling factors \( e^{-\lambda_d(i-j)} \) and \( c'(w_j)^p_h \) are constant with respect to the specific word at the \( i \)th position, and their combined contribution can be seen in Figure 4.1, where the words \textit{cooked} and \textit{fish} contribute the most to the generation of \( w_i \) under
Figure 4.1: Product of two scaling factors (green) from $e^{-0.2(i-j)}$ (yellow) and $c'(w_j)^{-0.5}$ (blue), where $c'(w_j)$ comes from the first 300 million words of the English Gigaword-4 corpus.

$P_l(w_i|h)$. $P_l(w_i|h)$ is used in conjunction with $P_C(C_i|C_h)$ to ensure that the generated words are not only lexically congruent with the history, but also that they are members of word classes that have a high likelihood given $C_{i-m+1} \ldots C_{i-1}$.

In addition to findings from cognitive neuroscience, this language model component is also supported by psycholinguistic research. The manner in which sentence context $(h)$ has been shown to influence lexical processing sustains our approach to lexical integration. Written target words are processed faster in both lexical decision tasks and eye-tracking experiments when these words are lexically related to the sentence context (Ehrlich and Rayner, 1981; Stanovich and West, 1983; Balota et al., 1985; Simpson et al., 1989; Hess et al., 1995). Priming by lexical association has been shown to be fast and automatic, in contrast to attentional priming, which requires active expectation or inhibition of a target word (Posner and Snyder, 1975; Schneider and
Furthermore, target word facilitation occurs in a manner commensurate with how congruent it is, in terms of its lexical semantics, with its context (Tweedy et al., 1977; den Heyer et al., 1983; den Heyer, 1985). Also, our decision to scale lexical influence in the history by a decaying distance is supported by evidence by Carroll and Slowiaczek (1986).

We evaluated numerous association measures—thirteen in all, which are listed in Appendix B. Pilot experiments showed that the Jaccard index and conditional probability gave the lowest perplexity, with Yule’s coefficient and Dice’s coefficient also performing well. Combining multiple association measures made a negligible improvement over the top-performing individual ones. We also found that the two rightmost terms in Equation 4.5—the history frequency penalty and the rare word penalty—made negligible contributions as well.

These two findings about the performance of various configurations of Equation 4.5 can be used to our advantage. We can reformulate this equation to reflect the generative nature of this model, without the need of costly normalization. The normalization term in that equation was originally employed to allow for the use of a variety of association measures without the need to reformulate the model for each measure. But by using one of the top-performing association measures, conditional probability, we can revise the equation as:

\[
P_{t_2}(w_i | w_0 \ldots w_{i-1}) \triangleq \lambda_u |V|^{-1} + (1 - \lambda_u) \sum_{j=0}^{i-1} P_d(w_i | w_j) \sum_{k=0}^{i-1} e^{-\lambda_d (i-k)}
\]

Where \(\lambda_u\) is an interpolation coefficient used for handling unknown predicted words,
$|V|$ is the cardinality of the vocabulary, $\lambda_d$ is a decay constant, and $P_d$ is described below. Relative to Equation 4.5, we locally normalized the decay coefficient, and interpolated with $|V|^{-1}$. Note that the calculation of the decay terms and their normalization (the far right fraction in Formula 4.6) can be easily precomputed and stored in a small static matrix of $n \times n$ cells, where $n$ is the maximum history length.

We first present a naïve version of $P_d$, which does not account for the distance from $i$ to $j$:

$$P_{d_0}(w_i|w_j) \triangleq c(w_j)^{-1} \sum_{w \in W} \sum_{f=j+1}^{|w|} \delta_{w_iw_f}(|w| - f)^{-1}$$  \hspace{1cm} (4.7)

Where $W$ is the set of sentences in the training set, $|w|$ is the length of a given sentence, and $\delta$ is the Kronecker delta, an indicator function where the predicate is equality. We can generalize Formula 4.7 to allow for a diminishing coefficient of the inner term as $i - j$ increases:

$$P_{d_1}(w_i|w_j) \triangleq c(w_j)^{-1} \sum_{w \in W} \sum_{f=j+1}^{|w|} \delta_{w_iw_f} \frac{e^{-\lambda_d(f-j)}}{\sum_{k=j+1}^{|w|} e^{-\lambda_d(k-j)}}$$  \hspace{1cm} (4.8)

Where $\lambda_d$ is a decay constant, which if set to 0 is equivalent to Formula 4.7.

Furthermore, if $\lambda_d$ is set to $\infty$, it is equivalent to a traditional bigram model. We illustrate varied values of $\lambda_d$ in Figure 4.2. While we use a normalized exponential decay here, because the span from $j + 1$ to $|w|$ is a bounded interval, we may use any
arbitrarily-defined normalized decay function. For the purposes of efficiency, we may wish to limit the future $f$ up to a set length $F$ that may be shorter than $|w| - j$. The specific future length $\mu$ for a given span is defined piecewise as:

\[
\mu = \begin{cases} 
|w| - j & \text{if } j + F > |w| - 1 \\
F & \text{otherwise}
\end{cases}
\]  

(4.9)

Which would give us:

\[
P_{d_2}(w_i|w_j) \triangleq c(w_j)^{-1} \sum_{w \in W} \sum_{f=j+1}^{j+\mu} \delta_{w_f w_i} e^{-\lambda_d(f-j)} \sum_{k=j+1}^{j+\mu} e^{-\lambda_d(k-j)}
\]

(4.10)

We are not restricted to generalizing bigram models. We can replace $w_j$ in the above formula with a span of any order $n$ such that the rightmost word in that span is $j$, represented as $w_j$:

\[
P_{d}(w_i|w_j) \triangleq c(w_j)^{-1} \sum_{w \in W} \sum_{f=j+1}^{j+\mu} \delta_{w_f w_i} e^{-\lambda_d(f-j)} \sum_{k=j+1}^{j+\mu} e^{-\lambda_d(k-j)}
\]

(4.11)

Just as Formula 4.10 is a generalization of bigram language models, Formula 4.11 is a generalization of $n$-gram language models. Thus setting $\lambda_d$ to $\infty$, it is equivalent to a traditional $n$-gram model. We may accordingly revise Formula 4.6 to support arbitrary $n$-grams, as:
\[ P_{\ell}(w_i|w_0 \ldots w_{i-1}) \triangleq \lambda_0|V|^{-1} + \lambda_1 P_{\text{ML}}(w_i) + \lambda_2 \sum_{j=0}^{i-1} P_d(w_i|w_j) \frac{e^{-\lambda_d(i-j)}}{\sum_{k=0}^{i-1} e^{-\lambda_d(i-k)}} 
\]
\[ \vdots \]
\[ + \lambda_n \sum_{j=0}^{i-1} P_d(w_i|w_j) \frac{e^{-\lambda_d(i-j)}}{\sum_{k=0}^{i-1} e^{-\lambda_d(i-k)}} \]

where \( \lambda_0 \ldots \lambda_n \) observe the following properties:

\[ \sum_{n} \lambda_n = 1 \]

\[ \bigwedge_{n} 0 \leq \lambda_n \leq 1 \]

We refer to Equation 4.12 as a decaying language model. The unigram term \( P_{\text{ML}}(w_i) \) in Equation 4.12 is largely optional for all but the smallest training sets. There is much less need for such a term in decaying language models than in \( n \)-gram models because the history in the former is not contiguous. We have observed that the weight for this term using medium and large training sets is usually tuned to zero.

4.3.1 Illustrations

Let us look at \( P_d \) (Equation 4.11) on an example training sentence fragment, “... cooked the fish in a pan”. At around the word “cooked”, a bigram model training algorithm
would increment the count of this word and the count of the bigram “cooked the”. From the perspective of a decaying language model, we would say that all of the (normalized) future probability mass for the word “cooked” is allocated to the word “the”, which is what happens if the future probability mass decays infinitely quickly. This is illustrated in the bottom chart of Figure 4.2.

However, the future probability may decay more slowly, and some of it may be allocated to subsequent words. Thus instead of incrementing the bigram “cooked the” by exactly one, the model could increment it by a value in the range of a unit interval \( I = [0, 1] \). Regardless of how quickly it decays, we ensure that all future probability mass sums to one by the denominator in the far right term of Equation 4.11. Thus even if there is no decay at all (i.e. \( \lambda_d = 0 \)) as in the top chart of Figure 4.2, or more pathologically if \( \lambda_d = -\infty \), we can still efficiently train a model by simply incrementing \( h \) by 1 and a word in the future \( w \in \mu \) by \( i \in I \).

We can also view \( P_d \) as a dynamic Bayesian network in Figure 4.3, where \( w_0 \) corresponds to the start-of-sentence token \(<s>\) and \( w_k \) the end-of-sentence token \( </s>\). The thickness of the edges corresponds to future probability allocation. Here we display extended bigrams in the top edges and extended trigrams in the bottom edges, and we set \( \lambda_d = 0.4 \) for both in this figure, based on the average parameter estimations of several languages and training set sizes. Note that the outgoing edge thickness should be the same at all vertices (before potentially getting split up), but are not displayed as such due to limitations in the drawing software.

By setting \( \lambda_d = \infty \) we have an interpolated trigram model, as is shown in Figure 4.4. Graphically this has the effect of eliminating all but the closest edges. The remaining
Figure 4.2: Future probability mass allocation for $P_d$ at ‘cooked’, with varying decay constants.
Figure 4.3: Dynamic Bayesian Network of a decaying language model’s \( P_d \), using extended bigrams (top edges) and extended trigrams (bottom edges). Based on experimental results \( \lambda_d = 0.4 \) for both sets of edges in this illustration.

Figure 4.4: Dynamic Bayesian Network of an interpolated \( n \)-gram language model (Jelinek and Mercer, 1980), using bigrams (top edges) and trigrams (bottom edges). Edges have the same thickness, corresponding to 100% future probability allocation into a single edge.

In the context of informal spoken conversations, decaying language models allow for more flexibility in handling disfluencies and filler words. For example, an \( n \)-gram model would need training instances of “public um school”, in order to provide a high probability of “school” given a history of “public um”. A skip language model would need training instances of “public \( w \) school”, where \( w \in V \cup \text{<unk>} \). On the other hand, a decaying language model can make use of training instances of “public school”.

55
Table 4.1: Examples from the Switchboard development set, with relative improvements of using a decaying language model over using a 4-gram modified Kneser-Ney model. The greyed text in the predicted word column is additional context for the reader (of this dissertation).

Table 4.1 presents some example sentence fragments that illustrate this flexibility on spoken conversational data in the Switchboard corpus v1r2 (Godfrey and Holliman, 1997). The sentences come from the development set, with the right column giving the relative improvement of using a decaying language model over a 4-gram interpolated modified Kneser-Ney language model (Chen and Goodman, 1998).

We can also look at the words having the highest probability of being predicted given the history of “BECAUSE I WENT THROUGH THE PUBLIC UM”, which occurred in the Switchboard development set.\(^1\) Table 4.2 shows the highest probability words using either a 4-gram modified Kneser-Ney model (left column) or a decaying language model (right column), trained on the training section of the Switchboard corpus (Godfrey and Holliman, 1997). The probabilities are shown in a log\(_{10}\) scale, as is common in language modeling software. The transcriptions for the Switchboard corpus

\(^1\)The decaying language model tunes eight dense, global parameters on the development set, so this specific sentence will have a negligible effect on the results presented in this paragraph.
have all uppercase text, so all relevant experiments and queries are accordingly all uppercase.

The realized word “school” was ranked 241\textsuperscript{st} by the 4-gram modified Kneser-Ney model (having a probability of 0.0001996), while it was ranked 6\textsuperscript{th} by the decaying language model (with a probability of 0.02409). The word “school” was ranked as high as it was in the \textit{n-gram} model (241\textsuperscript{st}) because of a single occurrence of “um school” in the training set, which is incidental—the distribution of words following “um” is close to the unigram distribution. Otherwise it would have been ranked 295\textsuperscript{th} (0.0001351) while the decaying language model would have kept “school” at 4\textsuperscript{th} place with almost no change in probability (0.01761). This shows that decaying language models are robust in the presence of disfluencies and filler words, even commonly-occurring ones like “um” or “you know”.

Figure 4.5 shows the full probabilities of predicted words given the history “\textit{because i went through the public um}”, using either a 4-gram interpolated modified Kneser-Ney \textit{n-gram} model or a decaying model. Both models are trained on the training set of the Switchboard corpus v1r2. The overall distribution of predicted words is smoother for the decaying language model than the \textit{n-gram} model. This is not surprising given that histories in the former model are composed of decaying word and \textit{n-gram} influences, whereas the latter model has a small set of contiguous histories, almost all of which include \( w_{i-1} \).

This previous word \( w_{i-1} \) can have a tremendously negative effect on \textit{n-gram} probabilities of \( w_i \) in four situations. Firstly if \( w_{i-1} \) is unseen in the training set, the \textit{n-gram} will necessarily have to resort to using unigram and uniform distributions.
<table>
<thead>
<tr>
<th>MKN-4 Predicted Word</th>
<th>Probability</th>
<th>DKLM Predicted Word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>UH</td>
<td>-0.621939</td>
<td>&lt;/s&gt;</td>
<td>-0.751262</td>
</tr>
<tr>
<td>&lt;/s&gt;</td>
<td>-0.902492</td>
<td>UH</td>
<td>-1.085735</td>
</tr>
<tr>
<td>I</td>
<td>-1.43137</td>
<td>I</td>
<td>-1.456835</td>
</tr>
<tr>
<td>YOU</td>
<td>-1.59963</td>
<td>THE</td>
<td>-1.592475</td>
</tr>
<tr>
<td>AND</td>
<td>-1.64739</td>
<td>AND</td>
<td>-1.608133</td>
</tr>
<tr>
<td>THE</td>
<td>-1.6977</td>
<td>SCHOOL</td>
<td>-1.618164</td>
</tr>
<tr>
<td>A</td>
<td>-1.91432</td>
<td>OUTRAGE</td>
<td>-1.647968</td>
</tr>
<tr>
<td>WE</td>
<td>-1.92514</td>
<td>SCHOOLS</td>
<td>-1.795641</td>
</tr>
<tr>
<td>IN</td>
<td>-1.94365</td>
<td>YOU</td>
<td>-1.842811</td>
</tr>
<tr>
<td>BUT</td>
<td>-1.96008</td>
<td>THAT</td>
<td>-1.851889</td>
</tr>
<tr>
<td>THEY</td>
<td>-2.00964</td>
<td>A</td>
<td>-1.878146</td>
</tr>
<tr>
<td>IT</td>
<td>-2.01788</td>
<td>SERVICE</td>
<td>-1.885289</td>
</tr>
<tr>
<td>WELL</td>
<td>-2.01863</td>
<td>TO</td>
<td>-2.002438</td>
</tr>
<tr>
<td>MY</td>
<td>-2.03704</td>
<td>IT</td>
<td>-2.080819</td>
</tr>
<tr>
<td>WHAT</td>
<td>-2.06675</td>
<td>RADIO</td>
<td>-2.119594</td>
</tr>
<tr>
<td>SO</td>
<td>-2.12715</td>
<td>SYSTEM</td>
<td>-2.121837</td>
</tr>
<tr>
<td>I'M</td>
<td>-2.1327</td>
<td>IN</td>
<td>-2.144550</td>
</tr>
<tr>
<td>THAT</td>
<td>-2.16646</td>
<td>OF</td>
<td>-2.174029</td>
</tr>
<tr>
<td>IT'S</td>
<td>-2.17429</td>
<td>EDUCATION</td>
<td>-2.208815</td>
</tr>
<tr>
<td>UM</td>
<td>-2.23538</td>
<td>KNOW</td>
<td>-2.213931</td>
</tr>
<tr>
<td>LIKE</td>
<td>-2.249</td>
<td>BUT</td>
<td>-2.214044</td>
</tr>
<tr>
<td>YEAH</td>
<td>-2.25626</td>
<td>IS</td>
<td>-2.219998</td>
</tr>
<tr>
<td>JUST</td>
<td>-2.28212</td>
<td>WE</td>
<td>-2.261798</td>
</tr>
<tr>
<td>FOR</td>
<td>-2.29461</td>
<td>THEY</td>
<td>-2.295305</td>
</tr>
<tr>
<td>SOME</td>
<td>-2.29619</td>
<td>IT'S</td>
<td>-2.298173</td>
</tr>
<tr>
<td>I-</td>
<td>-2.30833</td>
<td>LIKE</td>
<td>-2.329801</td>
</tr>
<tr>
<td>WHEN</td>
<td>-2.31938</td>
<td>WELL</td>
<td>-2.347677</td>
</tr>
<tr>
<td>IF</td>
<td>-2.32108</td>
<td>SO</td>
<td>-2.354199</td>
</tr>
<tr>
<td>I'VE</td>
<td>-2.32832</td>
<td>YEAH</td>
<td>-2.355075</td>
</tr>
<tr>
<td>ON</td>
<td>-2.33806</td>
<td>WAS</td>
<td>-2.368533</td>
</tr>
<tr>
<td>WITH</td>
<td>-2.34939</td>
<td>UM</td>
<td>-2.370866</td>
</tr>
<tr>
<td>TO</td>
<td>-2.35416</td>
<td>ARE</td>
<td>-2.419954</td>
</tr>
<tr>
<td>OH</td>
<td>-2.35688</td>
<td>HAVE</td>
<td>-2.440259</td>
</tr>
<tr>
<td>HE</td>
<td>-2.36722</td>
<td>THINK</td>
<td>-2.460921</td>
</tr>
<tr>
<td>AT</td>
<td>-2.39786</td>
<td>MY</td>
<td>-2.465227</td>
</tr>
<tr>
<td>THERE</td>
<td>-2.39902</td>
<td>WHAT</td>
<td>-2.469228</td>
</tr>
<tr>
<td>ONE</td>
<td>-2.40538</td>
<td>FOR</td>
<td>-2.469912</td>
</tr>
<tr>
<td>AS</td>
<td>-2.41064</td>
<td>DON'T</td>
<td>-2.475793</td>
</tr>
<tr>
<td>THAT'S</td>
<td>-2.4265</td>
<td>JUST</td>
<td>-2.505816</td>
</tr>
<tr>
<td>THIS</td>
<td>-2.43416</td>
<td>WHOLE</td>
<td>-2.513273</td>
</tr>
</tbody>
</table>

Table 4.2: Top 40 predicted words given the history “BECAUSE I WENT THROUGH THE PUBLIC UM”, trained on the training section of the Switchboard corpus. Probabilities are in a log_{10} scale.
Figure 4.5: Ranked probabilities of predicted words in the Switchboard corpus, given the history “BECAUSE I WENT THROUGH THE PUBLIC UM”. Both the $x$ and $y$ axes are in a $\log_{10}$ scale.
Secondly, if $w_{i-1}$ occurs a small handful of times, then it will have a very small set of words following it in the training set, increasing the reliance of $n$-gram models on unigram and uniform distributions. Thirdly, even if $w_{i-1}$ occurs frequently, if it occurs before a wide variety of words in the training set, as is the case with “um” and “you know”. That is, if $H(w|w_{i-1}) \gg 0$, regardless of $p(w_{i-1})$. Lastly, if the training set domain differs greatly from the test set domain with respect to $w_{i-1}$ ($P_{\text{train}}(w_i|w_{i-1}) \neq P_{\text{test}}(w_i|w_{i-1})$), as we shall see below.

The Switchboard corpus is quite small, containing less than a million training words. It is quite reasonable to use this small specialized corpus in conjunction with a much larger, more general corpus. We performed the same analysis using 100M training words from the English Gigaword-4 corpus (Parker et al., 2009), and show the results in Table 4.3. The text for this corpus have been converted to all lowercase, so all relevant experiments and queries are accordingly all lowercase.

The results for the 4-gram modified Kneser-Ney are also bad here, but for very different reasons. The word “um” occurs in this corpus, but is exclusively used for non-English words. The primary use is the Arabic word for “mother”, which is often used as a teknonym (eg. *Umm Kulthum* for the mother of *Kulthum*), which can become a place name (eg. *Umm Qais*). This word is also used in proper nouns in Khmer (*Um Chien, Um Sarith*), Karen (*Um Phang*), and Korean (*Um Rank-Yong, Um Man-Kyu*). After these words, words having high unigram probability occur, which is why “school” is ranked 887th by the 4-gram modified Kneser-Ney model. The results from this model are expected, since using an out-of-domain corpus naturally leads to lexical

---

2Both the interpolated probability in this context and the unigram probability are about 7.5e-05.
<table>
<thead>
<tr>
<th>MKN-4 Predicted Word</th>
<th>Probability</th>
<th>DKLM Predicted Word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>kalsum</td>
<td>-1.20603</td>
<td>,</td>
<td>-1.351352</td>
</tr>
<tr>
<td>qasr</td>
<td>-1.34694</td>
<td>the</td>
<td>-1.356809</td>
</tr>
<tr>
<td>chien</td>
<td>-1.55681</td>
<td></td>
<td>-1.448510</td>
</tr>
<tr>
<td>tuba</td>
<td>-1.56585</td>
<td>to</td>
<td>-1.573603</td>
</tr>
<tr>
<td></td>
<td>-1.56954</td>
<td>and</td>
<td>-1.626755</td>
</tr>
<tr>
<td>said</td>
<td>-1.72157</td>
<td>of</td>
<td>-1.670305</td>
</tr>
<tr>
<td>'s</td>
<td>-1.75189</td>
<td>sector</td>
<td>-1.725918</td>
</tr>
<tr>
<td>mohammad</td>
<td>-1.79824</td>
<td>a</td>
<td>-1.778515</td>
</tr>
<tr>
<td>jihad</td>
<td>-1.79927</td>
<td>in</td>
<td>-1.876382</td>
</tr>
<tr>
<td>salah</td>
<td>-1.79938</td>
<td>&lt;/s&gt;</td>
<td>-1.890744</td>
</tr>
<tr>
<td>shaker</td>
<td>-1.79956</td>
<td>'s</td>
<td>-1.910984</td>
</tr>
<tr>
<td>sarith</td>
<td>-1.79963</td>
<td>&quot;</td>
<td>-1.985086</td>
</tr>
<tr>
<td>sagata</td>
<td>-1.79964</td>
<td>deficit</td>
<td>-1.998472</td>
</tr>
<tr>
<td>phang</td>
<td>-1.79964</td>
<td>prosecutor</td>
<td>-2.017389</td>
</tr>
<tr>
<td>al-maradim</td>
<td>-1.79964</td>
<td>on</td>
<td>-2.080709</td>
</tr>
<tr>
<td>al-maarik</td>
<td>-1.79964</td>
<td>for</td>
<td>-2.149420</td>
</tr>
<tr>
<td></td>
<td>-2.24152</td>
<td>security</td>
<td>-2.169206</td>
</tr>
<tr>
<td>...</td>
<td>-2.27013</td>
<td>interest</td>
<td>-2.237485</td>
</tr>
<tr>
<td>and</td>
<td>-2.27205</td>
<td>was</td>
<td>-2.247535</td>
</tr>
<tr>
<td>mahmud</td>
<td>-2.3071</td>
<td>eye</td>
<td>-2.297888</td>
</tr>
<tr>
<td>nasser</td>
<td>-2.30731</td>
<td>that</td>
<td>-2.299482</td>
</tr>
<tr>
<td>hashem</td>
<td>-2.30813</td>
<td>is</td>
<td>-2.328236</td>
</tr>
<tr>
<td>ammar</td>
<td>-2.30825</td>
<td>with</td>
<td>-2.334055</td>
</tr>
<tr>
<td>iyad</td>
<td>-2.30828</td>
<td>streets</td>
<td>-2.365032</td>
</tr>
<tr>
<td>qais</td>
<td>-2.30836</td>
<td>qasr</td>
<td>-2.414991</td>
</tr>
<tr>
<td>kasr</td>
<td>-2.30841</td>
<td>will</td>
<td>-2.415145</td>
</tr>
<tr>
<td>sahira</td>
<td>-2.30843</td>
<td>service</td>
<td>-2.424622</td>
</tr>
<tr>
<td>rank-yong</td>
<td>-2.30843</td>
<td>at</td>
<td>-2.425382</td>
</tr>
<tr>
<td>qeis</td>
<td>-2.30843</td>
<td>health</td>
<td>-2.434705</td>
</tr>
<tr>
<td>naqa</td>
<td>-2.30843</td>
<td>as</td>
<td>-2.451144</td>
</tr>
<tr>
<td>man-kyu</td>
<td>-2.30843</td>
<td>...</td>
<td>-2.468122</td>
</tr>
<tr>
<td>kalthum</td>
<td>-2.30843</td>
<td>by</td>
<td>-2.473350</td>
</tr>
<tr>
<td>kalthoum</td>
<td>-2.30843</td>
<td>it</td>
<td>-2.497668</td>
</tr>
<tr>
<td>durman</td>
<td>-2.30843</td>
<td>first</td>
<td>-2.510066</td>
</tr>
<tr>
<td>denizcilik</td>
<td>-2.30843</td>
<td>said</td>
<td>-2.520389</td>
</tr>
<tr>
<td>al-qiwain</td>
<td>-2.30843</td>
<td>would</td>
<td>-2.526126</td>
</tr>
<tr>
<td>al-hassam</td>
<td>-2.30843</td>
<td>has</td>
<td>-2.554091</td>
</tr>
<tr>
<td>in</td>
<td>-2.40084</td>
<td>night</td>
<td>-2.555335</td>
</tr>
<tr>
<td>of</td>
<td>-2.53691</td>
<td>his</td>
<td>-2.556341</td>
</tr>
<tr>
<td>to</td>
<td>-2.54218</td>
<td>kalsum</td>
<td>-2.574861</td>
</tr>
</tbody>
</table>

Table 4.3: Top 40 predicted words given the history “because i went through the public um”, trained on 100 million words of the English Gigaword-4 corpus.
(and structural) mismatches. Furthermore, since \(n\)-gram models use a contiguous history, well-attested intervening words like “you know” will still have deleterious effects.

The top predicted words by the decaying language model are in line with terminology relating to governmental words—public sector, public deficit, public prosecutor, etc. The next few hundred words are filled with other government-related words. The word “school” appears at 395, having 4.5 times greater probability than with the 4-gram modified Kneser-Ney model. The drop in rank for “school” compared to the Switchboard-trained decaying model is not surprising, given that news tends to focus more on financial and legal stories, while spoken dialogues focus more on personal events. The decaying language model is also more shielded from the detrimental effects of “um”, with the top \(n\)-gram word “kalsum” appearing at 40 here.

### 4.4 P600-based Language Model Component

While the ELAN and its statistical language model counterpart are concerned with local morpho-syntactic processing, and the N400 with longer-distance lexical semantics, we would like to have a component dealing with sentence-level structural processing based on our knowledge of the P600.

The P600 can help inform us as to how sentence-level syntactic structure is built-up neurophysiologically. Various characteristics of the P600 described in section 2.3 include the following:

- It occurs in the presence of subjacency/ECP violations
• It occurs in verb argument mismatches, including both incorrect number of verb arguments and type of verb arguments

• Grammatical features of words, such as number & gender, are integrated into structure building

• Online grammatical reanalysis also elicits a P600

• Well-formed, unambiguous long-distance dependencies induce a P600

Implementing a faithful statistical language model component based on these characteristics is a thorny endeavor. Furthermore, previous syntactic language models have shown little gains to be had in experiments involving speech recognition or machine translation (cf. Baker, 1979; Chen, 1995; Chelba, 2000; Roark, 2001, 2002; Schwartz et al., 2011, inter alia). In the future, we would like to develop a P600 statistical language model component that allows for incremental decoding, and needs no manually-annotated treebank for training. One interesting starting point is the dependency-style syntactic language model of Gao and Suzuki (2003). The joint probability $P(w_1 \ldots w_n, D)$ in this model is defined:

$$P_s(w_1 \ldots w_n, D) \triangleq \prod_{i=1}^{n} P(w_i|\Phi(w_1 \ldots w_{i-1}, D_{i-1})P(D|w_1 \ldots w_{i-1}) \quad (4.13)$$
where $D_j$ is a dependency structure covering $w_1 \ldots w_j$, and $\Phi$ is a mapping function from $(h, D)$ to equivalence classes. The probability of $P(w_i|h)$ is defined as:

$$P_s(w_i|\Phi(w_1 \ldots w_{i-1}, D_{i-1}) \triangleq P(H_i|\Phi(w_1 \ldots w_{i-1}, D_{i-1}) \times P(w_i|\Phi(w_1 \ldots w_{i-1}, D_{i-1}), H_i)$$

$$+ P(F_i|\Phi(w_1 \ldots w_{i-1}, D_{i-1}) \times P(w_i|\Phi(w_1 \ldots w_{i-1}, D_{i-1}), F_i)$$

where $H_i$ and $F_i$ are the category of the headword or dependent (respectively) of the sentence prefix at the $i^{th}$ position (a binary classification).

The probability of $P(D|w_1 \ldots w_{i-1})$ defined as:

$$P(D|w_1 \ldots w_{i-1}) \approx \prod_{d \in D} P(d)$$

This approximation is made, using $P(d)$ rather than $P(d|w_1 \ldots w_{i-1})$, since it is unlikely that $w_1 \ldots w_{i-1} \in W$, where $W$ is a training set, for all but the shortest histories. As for $P(d)$, this is defined as:

$$P(d_{ij}) = \frac{c(w_i, w_j, R)}{c(w_i, w_j)}$$

where $c(w_i, w_j, R)$ is the number of times there exists a dependency relation $R$ between $w_i$ and $w_j$ in the training set. The denominator represents the number of sentences in the training set containing both $w_i$ and $w_j$, and $R$ is induced using an EM-style
algorithm reminiscent to the one used in Klein and Manning (2004, p. 482), although with a cruder initialization, and a more refined distinction of heads and dependents.
Chapter 5

Experiments

5.1 Setup

We seek to not only evaluate whether our model is effective, but in what conditions is it effective. For example, Brants et al. (2007) found that Kneser-Ney smoothed \( n \)-gram language models (Kneser and Ney, 1995) outperformed the simpler Stupid Backoff \( n \)-gram models when trained on smaller data sets, but they performed equally well on very large training sets. They also saw enormous differences in perplexity between smaller in-domain Kneser-Ney smoothed models and larger out-of-domain ones, with the smaller in-domain models having lower perplexity than out-of-domain models ten times larger. Interestingly, higher perplexity did not translate to lower BLEU scores in their Arabic-English machine translation task. These are examples in experimental setup that help identify strengths and weaknesses in particular approaches to using statistical language models.

We evaluate our model in textual and speech recognition experiments. In the future we’d like to conduct machine translation experiments as well. Modern speech recognition systems share many similarities to modern statistical machine translation systems, but there are important strengths and weaknesses in each for the purposes of
evaluating a language model. The task of machine translation is inherently multilingual, so there are many more languages on which to evaluate a language model. However, the task of speech recognition does not require reordering like machine translation does, which complicates evaluation. Furthermore, there is reason to believe that the N400-based component of our language model can benefit speech recognition more than machine translation. This component makes use of a long history in helping select the predicted word. Machine translation systems can use translation models to help in lexical selection, making use of language models for fluency. Speech recognition systems, on the other hand, do not need to ensure long-distance grammatical fluency as much, but rather long-distance lexical coherency. This is exactly what the N400-based component ensures. The specifics of each of these experiments are discussed in Sections 5.4 and 5.5.

Within each of these tasks we compare our model with \(n\)-gram models as the size of the training set increases. Specifically, we start with \(10^6 = 1\) million tokens, incrementing the exponent until each training set is exhausted.

In the textual experiments we evaluate our model on multiple languages. We look in particular at languages having a freer word order and richer morphology, which pose challenges for language models. In light of the disfluency examples illustrated in Section 4.3.1, we use an informal conversational corpus for our speech recognition experiments.
5.2 Implementation

“I mean, if 10 years from now, when you are doing something quick and dirty, you suddenly visualize that I am looking over your shoulders and say to yourself ‘Dijkstra would not have liked this’, well, that would be enough immortality for me.”

–Edsger Dijkstra

We have implemented our model in C (cf. Kernighan and Ritchie, 1988; ISO/IEC JTC1/SC22/WG14, 1999) to allow for speed and portability. The code is parallelized using OpenMP (OpenMP ARB, 2013). We jointly tune all parameters by optimizing perplexity on held-out data, using a parallelized direct search method, due to the high cost of evaluating the objective function. Using the 1B word English Gigaword model, the system was able to process 260K test words per second.

The modified Kneser-Ney training and querying software comes from KenLM (Heafield, 2011; Heafield et al., 2013), which is a fast, lightweight, and scalable set of programs written in C++. This software is included in popular machine translation decoders like Moses (Koehn et al., 2007), cdec (Dyer et al., 2010), and Joshua (Post et al., 2013). KenLM can either store the language model as a probing hash table or a trie data structure. Hash tables are faster than tries, but use more memory and disk space. Due to the large number of experiments, we store binarized n-gram models as tries.

The word class induction software, mkcls (Och, 1999), uses the exchange algorithm (Kneser and Ney, 1993; Martin et al., 1998), and is also distributed with Moses. The software by Clark (2003) which extends the exchange algorithm to use orthotactic
information has severe scalability limitations. Even so, the exchange algorithm itself still has limitations for inducing a large number of word classes, and thus we use 200 and 400 word classes in our experiments, using the first 20K lines of training data.

The code for conducting the textual experiments is implemented as a Makefile, to ensure that all experiments are reproducible, and that they require only as many computational and disk resources as are required for a given experiment. We report the time required for training and decoding, since these are important considerations for large-scale implementations.

5.3 Evaluation Measures

5.3.1 Perplexity

Perplexity (Bahl et al., 1983) is a measurement of the average number of words being predicted, given a context (history):

\[ PP = 2^{\frac{1}{n} \sum_{i=1}^{n} \log_2 p(w_i|w_1 \cdots w_{i-1})} \]  

Cross-entropy is the binary logarithm of perplexity, and it represents here the average number of bits needed to encode a predicted word. A reduction of one bit in cross-entropy corresponds to a 50% reduction in perplexity. Thus perplexity comparisons are discussed in relative terms, while cross-entropy comparisons are discussed in absolute terms. A reduction in cross entropy from 9 bits to 8 bits is the
same relative improvement as a reduction from 5 bits to 4 bits, even though one has a 11% relative reduction in bits and another 20%.

Since statistical language models are usually incremental, it is common to include the end-of-sentence symbol \(<s>\) in the calculation of perplexity. We follow this convention in our experiments as well. The start-of-sentence symbol \(<s>\) is never predicted in an incremental setting, but rather is only used in histories. Hence it is not included in the tabulation of perplexity.

Closed vocabulary systems, which only train on and predict a predetermined number of vocabulary words (types), typically ignore out-of-vocabulary (OOV) words in determining perplexity. Open vocabulary systems, on the other hand, do reserve probability for such unseen words, and typically consider them in calculating perplexity. All of our experiments and models are open vocabulary.

For our experiments we use the same dedicated program for evaluating perplexity on the outputs of all models.

5.3.2 Word Error Rate

Word error rate (WER) is a measurement of how many word insertions, deletions, and substitutions are necessary for a hypothesis sentence to match a reference (correct) sentence, relative to the length of the reference sentence. It is based upon Levenshtein distance (Levenshtein, 1966), using words instead of characters as the basic unit. It is defined as:

\[
WER = \frac{i + d + s}{n_{\text{ref}}} \quad (5.2)
\]
where \( i \), \( d \), and \( s \) are the number of insertions, deletions, and substitutions respectively. The denominator, \( n_{\text{ref}} \), is the number of words (tokens, actually) in the reference sentence. Computing the word error rate for an entire set of hypothesis sentences is similar, namely the sum of all word insertions, deletions, and substitutions, divided by the total number of words in the set of reference sentences.

Word error rate is the most common measurement in evaluating speech recognition systems. Unlike in the calculation of perplexity (§ 5.3.1), word error rate does not consider the end-of-sentence symbol \(<s>\). For our experiments we use the same dedicated program provided by Kaldi for evaluating word error rates on all models.

Klakow and Peters (2002) looked at the relationship between perplexity and word error rate, and how closely correlated they are. This relationship is important because if the two measurements are highly correlated, then language model parameters can be estimated and novel language models can be evaluated in a manner that is easier, faster, and less specific to a given speech corpus.

They evaluated several language models, including \( n \)-gram, class, and cache models, with hundreds of configurations involving the use or absence of pruning, different smoothing strategies, and training on various sizes of subcorpora. Since cache models are adaptive—they change parameters based on the test set—there is an option in speech recognition experiments to use either a supervised setting (using hand-annotated, correct histories) or an unsupervised setting (using the previously-predicted words in the history). The unsupervised setting reflects actual usage in a real speech recognition setting, since annotating histories with correct words obviates the need for speech recognition in the first place. However, they use a supervised
They conducted two broad experiments. The first, smaller one used the Wall Street Journal (WSJ) corpus of 40M words with a closed vocabulary of 64K words. The test set consisted of a small set of 2200 words on the topic of Jackie Kennedy. They reranked lattices from a 1994 DARPA evaluation detailed in Dugast et al. (1995). The oracle, or best possible word error rate, for this lattice was 11%, with most systems achieving about 33%. The results of these experiments are displayed in Figure 5.1.

The second, larger set of experiments also used a closed vocabulary of 64K words, and had a much larger training set consisting of the North American Business News corpus (240M words: Stern, 1996), the North American News Text corpus (1.2B words: Graff, 1995), as well as the Switchboard corpus of spoken telephone conversations.
Figure 5.2: Perplexity and word error rate results of various language models on the Eval96 and Eval97 test sets (Klakow and Peters, 2002, pg. 26) (1.7M words: Godfrey and Holliman, 1997). They used a test set of the Broadcast News 1996 corpus (20K words), the 1997 version (33K words), as well as a concatenation of these two. They reranked lattices from Beyerlein et al. (1998). The lattices for the Eval96 test set had an oracle word error rate of 17%, with most systems averaging about 34%. For the Eval97 test set the lattice had an oracle word error rate of 15%, and most systems averaged about 30%. The results of these experiments are displayed in Figure 5.2.

They found that the relationship between the word error rate results and the perplexity results fit into their hypothesized equation:

\[ \text{WER} \approx b \text{PP}^a \]  \hspace{1cm} (5.3)
<table>
<thead>
<tr>
<th>Test Set</th>
<th>Correlation $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kennedy</td>
<td>0.993 ± 0.048</td>
</tr>
<tr>
<td>Eval97</td>
<td>0.980 ± 0.088</td>
</tr>
<tr>
<td>Eval96+97</td>
<td>0.978 ± 0.073</td>
</tr>
<tr>
<td>Eval96</td>
<td>0.965 ± 0.124</td>
</tr>
</tbody>
</table>

Table 5.1: Pearson’s correlation coefficients of perplexity and word error rate on various test sets of Klakow and Peters (2002, pg. 27)

where values of $b$ from their experiments range from 6 (Kennedy) to 16 (Eval96), and $a$ from 0.17 (Eval96) to 0.27 (Kennedy). They make the interesting observation that “the slope $a$ is smaller for tasks that are acoustically more challenging. Hence on those tasks larger reductions in PP are needed to obtain a given reduction in WER” (Klakow and Peters, 2002, pg. 26).

They also found that log(PP) (cross-entropy) is very highly linearly correlated with log(WER). Pearson’s correlation coefficients ranged from 0.965 on the Eval96 test set to 0.993 on the Kennedy test set. Table 5.1 lists these correlation coefficients.

Goodman (2001, pg. 38) also made a less formal comparison of perplexity and word error rate across qualitatively different language models. The training set consisted of the North American Business news corpus (Stern, 1996), with subcorpora of 100K, 1M, 10M and 284M words. The development and test set of 600 sentences each were taken from every 50 sentences from the above corpus, with no overlap between any sets. They had a closed vocabulary of approximately 60K words, with out-of-vocabulary words not included in perplexity calculations. The word error rate experiments were
Their results are illustrated in Figure 5.3. Note that the $x$ and $y$ axes are switched relative to the previous figures, they report cross-entropy instead of its base-2 exponent, perplexity, and they report word error rate instead of log(WER) like in Figure 5.1. There appears to be a slightly weaker correlation between perplexity and word error rate than those found in Klakow and Peters (2002), although the figure is zoomed in to a much tighter range (8.8–10% WER) than the previous figures (25–50% and 30–45% WER). Unfortunately no correlation coefficients nor other relevant specific values were reported in Goodman (2001).
We can approximate the values of $a$ and $b$ from Equation 5.3 to the data in Figure 5.3. The value of $a$ is approximately 0.294, and the value of $b$ is somewhere near 2.42–2.53. This places the test set that Goodman (2001) closer to the Kennedy test set used by Klakow and Peters (2002). We plot these four slopes in Figure 5.4, including the approximation of Goodman (2001)’s results, the Kennedy test set, and the Eval96 and Eval97 test sets of Klakow and Peters (2002). Based on the Eval96, Eval97, Kennedy, and Goodman test sets, we can observe a generally inverse relationship between $a$ and $b$. This could suggest that, while there is greater potential influence of language models on absolute word error rate for test sets having a high $b$ coefficient, language models can still have an influence in reducing the relative word error rate for test sets having a low $b$ coefficient.

5.3.3 Bleu Score

The evaluation measure most often used in machine translation is BLEU score (Papineni et al., 2002). This is defined as a (possibly weighted) average of the number of $n$-grams in a hypothesized translation found in reference translations (for precision), subject to length restrictions (for recall). The first half of this, $n$-gram precision, is defined as:

$$p_n = \frac{\sum_{\mathcal{C} \in \{\text{Candidates}\}} \sum_{n\text{-gram} \in \mathcal{C}} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{\mathcal{C}'} \sum_{n\text{-gram}' \in \mathcal{C}'} \text{Count}_{\text{clip}}(n\text{-gram}')},$$ (5.4)

where the common value of $n$ is 4. As this measures precision, it is easy to imagine a
Figure 5.4: Plots of four test set function approximations, using linear scales. Solid line segments indicate attested ranges.
bad translation that could achieve a high $n$-gram precision. If a reference translation
contains ten words, a hypothesized translation could be composed of just one word
which we are most confident of. This is illustrated below:

Source: María no dió una bofetada a la bruja verde
Ref:  Maria did not slap the green witch
Hyp:  witch

The hypothesized translation would receive perfect $n$-gram precision of 1. We clearly
would like to ensure that hypothesized translations also convey all the basic ideas found
in the source text. This is not a trivial task, as we would need manual alignments
and/or human judgments. Papineni et al. (2002) opt for a simpler solution—penalize
short translations. They define a brevity penalty as a piecewise function:

$$BP = \begin{cases} 
1 & \text{if } |c| > |r| \\
\exp(1-|r|/|c|) & |c| \leq |r|
\end{cases} \quad (5.5)$$

where $|r|$ is the length of the reference translation and $|c|$ is the length of the candidate
(hypothesized) translation. The combined score is the product of the brevity penalty
and the geometric mean of unigram, bigram, trigram, and 4-gram precision:

$$\text{BLEU} = BP \cdot \exp \left( \sum_{n=1}^{N} w_n \log p_n \right) \quad (5.6)$$
where the weighting factor $w_n$ is (usually) set uniformly to:

$$w_n = \frac{1}{N}$$

(5.7)

5.4 Textual Experiments

5.4.1 Data

We used datasets in four languages that each have a large corpus available:

English (EN): The English Gigaword-4 corpus (Parker et al., 2009), which primarily consists of news articles from the New York Times, AFP, AP, Xinhua, CNA, and the LA Times.

Arabic (AR): The Arabic Gigaword-5 corpus (Parker et al., 2011a), which consists of news articles from nine online newspapers.

Croatian (HR\footnote{Following ISO 639-1 convention; from Hrvatski. Note the wordplay in the corpus name.}): The HRWAC-1 corpus (Ljubešić and Erjavec, 2011), which consists of processed web documents.

Hungarian (HU): The Hungarian Webcorpus (Halácsy et al., 2004), which consists of processed web documents.

Each text corpus is then preprocessed in the same manner by the Makefile, to ensure consistency:

1. Paragraphs are split up so that each sentence is on a separate line.
2. Digits are conflated.

3. Duplicate lines are removed.

4. Lines are put in a randomized order.

5. Characters are lowercased.

6. Sentences are tokenized, using the Moses tokenizer.

7. Punctuation is normalized, so that eg. non-breaking spaces (U+00A0) become normal spaces (U+0020). We use the Moses punctuation normalization script for this.

8. The corpus is split into training, development, and test sets. Every 98 of the now randomized sentences goes into the training set, then one into the development set, then one into the test set.

9. Subcorpora of the training set are generated, starting with $10^6 = 1$ million tokens, incrementing the exponent until each training set is exhausted. The full sentence containing each milestone token is included.

Due to the length of time required by the word class induction software, we used the first 20K lines ($\approx 350K$–$650K$ words) of the training set for subsequent word class induction. Thus this word class subcorpus is a proper subset of the smallest subcorpus used for experiments.

Table 5.2 shows information about the full training sets for the four languages. Each language also had smaller training corpora of 1 million, 10 million, and 100 million tokens. Table 5.3 shows information about the test sets, which is the same set regardless of the training set size. The development sets were very similar in size to the test sets. Notice that the full Arabic corpus has roughly half the number of lines

\[ \text{If the full training set is less than } \approx 1.2x \text{ the size of the previous subcorpus, we do not evaluate the full training set. Some full training sets could not fit into memory, so we used the largest training set that could fit into memory. The largest full training set was the same size for all language models.} \]
Table 5.2: Full training set sizes of various textual corpora.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Tokens</th>
<th>Types</th>
<th>Tokens</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN Gigaword-4</td>
<td>1B</td>
<td>2.7M</td>
<td>370</td>
<td>26M</td>
</tr>
<tr>
<td>AR Gigaword-5</td>
<td>375M</td>
<td>2.8M</td>
<td>135</td>
<td>13M</td>
</tr>
<tr>
<td>HR WAC-1</td>
<td>550M</td>
<td>4.6M</td>
<td>120</td>
<td>26M</td>
</tr>
<tr>
<td>HU Webcorpus</td>
<td>460M</td>
<td>7.1M</td>
<td>65</td>
<td>25M</td>
</tr>
</tbody>
</table>

Table 5.3: Test set sizes of various textual corpora.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Tokens</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN Gigaword-4</td>
<td>30M</td>
<td>770K</td>
</tr>
<tr>
<td>AR Gigaword-5</td>
<td>10M</td>
<td>350K</td>
</tr>
<tr>
<td>HR WAC-1</td>
<td>9M</td>
<td>400K</td>
</tr>
<tr>
<td>HU Webcorpus</td>
<td>10M</td>
<td>520K</td>
</tr>
</tbody>
</table>

as the other full corpora. Arabic script characters usually occupy two bytes in UTF-8 encoding, so Arabic text requires more memory than most Latin script text.\(^3\)

5.4.2 Results and Analysis

We first evaluate neurophysiologically-inspired language models (\(\nu\phi\)-LM; § 4.1), then decaying language models (§ 4.3). Tables 5.4 and 5.5 show perplexity results of using exponentially increasing training sets, for English (EN), Arabic (AR), Croatian (HR), Hungarian, and to a lesser extent Croatian also use use two-byte characters in UTF-8 for diacriticized letters, but undiacidicriticized Latin script letters use one byte.\(^3\)

\(^3\) Hungarian, and to a lesser extent Croatian also use use two-byte characters in UTF-8 for diacriticized letters, but undiacidicriticized Latin script letters use one byte.
and Hungarian (HU). In the former table $|C| = 200$ word classes are used, and in the latter table $|C| = 400$. The second column in each table lists perplexity for 4-gram interpolated modified Kneser-Ney models, and the third column lists perplexity when linearly interpolated with a $\nu\phi$-LM. The final column shows the relative reduction in perplexity of the cell in the third column over the cell in the second column. The relative reduction in perplexity is calculated as:

$$\Delta_{PP} \triangleq 1 - \frac{PP_1}{PP_0}$$

where $PP_0$ here comes from the 4-gram modified Kneser-Ney language model, and $PP_1$ from adding a $\nu\phi$-LM.

The accompanying weights of these models are found in Figures 5.5 and 5.6. Compared with the subsequent results in Table 5.6, we see that the class-based model greatly contributes for all languages when the amount of training data is small. As the training set grows, the class-based model’s contribution wanes, so that by about $10^{8.5}$ training tokens its influence is all but gone. It maintains its contribution in English the longest, perhaps due to the more rigid word order. Also, having fewer classes (200) is more helpful at the smaller training sizes, but having more classes (400) extends the utility of class-based models for medium and large training sets.

We can see here that decaying language models (DKLM) maintain constant weights across training set sizes, while the class-based model’s weight is largely reallocated to the $n$-gram model at the largest sizes. This suggests a high degree of orthogonality between decaying language models and $n$-gram language models, but less so between
<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN 1M</td>
<td>251.39</td>
<td>217.36</td>
<td>13.54%</td>
</tr>
<tr>
<td>EN 10M</td>
<td>187.69</td>
<td>172.87</td>
<td>7.90%</td>
</tr>
<tr>
<td>EN 100M</td>
<td>126.96</td>
<td>121.03</td>
<td>4.67%</td>
</tr>
<tr>
<td>EN 1B</td>
<td>83.91</td>
<td>81.67</td>
<td>2.67%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR 1M</td>
<td>831.35</td>
<td>692.68</td>
<td>16.68%</td>
</tr>
<tr>
<td>AR 10M</td>
<td>678.63</td>
<td>612.08</td>
<td>9.81%</td>
</tr>
<tr>
<td>AR 100M</td>
<td>410.08</td>
<td>381.26</td>
<td>7.03%</td>
</tr>
<tr>
<td>AR 375M</td>
<td>286.17</td>
<td>269.50</td>
<td>5.83%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR 1M</td>
<td>566.25</td>
<td>487.04</td>
<td>13.99%</td>
</tr>
<tr>
<td>HR 10M</td>
<td>601.73</td>
<td>540.46</td>
<td>10.18%</td>
</tr>
<tr>
<td>HR 100M</td>
<td>432.82</td>
<td>397.42</td>
<td>8.18%</td>
</tr>
<tr>
<td>HR 550M</td>
<td>310.67</td>
<td>289.12</td>
<td>6.94%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>HU 1M</td>
<td>399.89</td>
<td>358.52</td>
<td>10.35%</td>
</tr>
<tr>
<td>HU 10M</td>
<td>536.66</td>
<td>497.15</td>
<td>7.36%</td>
</tr>
<tr>
<td>HU 100M</td>
<td>437.31</td>
<td>408.48</td>
<td>6.59%</td>
</tr>
<tr>
<td>HU 460M</td>
<td>307.69</td>
<td>288.89</td>
<td>6.11%</td>
</tr>
</tbody>
</table>

Table 5.4: Perplexities of test sets using 4-gram modified Kneser-Ney models, as well as adding a $\nu\phi$-LM with $|C| = 200$ classes.
Figure 5.5: Interpolation weights for English, Arabic, Croatian, and Hungarian corpora, using $|C| = 200$ classes in the class-based model.
<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN 1M</td>
<td>251.39</td>
<td>218.08</td>
<td>13.25%</td>
</tr>
<tr>
<td>EN 10M</td>
<td>187.69</td>
<td>172.83</td>
<td>7.92%</td>
</tr>
<tr>
<td>EN 100M</td>
<td>126.96</td>
<td>120.58</td>
<td>5.03%</td>
</tr>
<tr>
<td>EN 1B</td>
<td>83.91</td>
<td>81.30</td>
<td>3.11%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR 1M</td>
<td>831.35</td>
<td>692.27</td>
<td>16.73%</td>
</tr>
<tr>
<td>AR 10M</td>
<td>678.63</td>
<td>611.43</td>
<td>9.90%</td>
</tr>
<tr>
<td>AR 100M</td>
<td>410.08</td>
<td>380.48</td>
<td>7.22%</td>
</tr>
<tr>
<td>AR 375M</td>
<td>286.17</td>
<td>269.43</td>
<td>5.85%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR 1M</td>
<td>566.25</td>
<td>486.59</td>
<td>14.07%</td>
</tr>
<tr>
<td>HR 10M</td>
<td>601.73</td>
<td>539.93</td>
<td>10.27%</td>
</tr>
<tr>
<td>HR 100M</td>
<td>432.82</td>
<td>396.36</td>
<td>8.42%</td>
</tr>
<tr>
<td>HR 550M</td>
<td>310.67</td>
<td>289.12</td>
<td>6.94%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>HU 1M</td>
<td>399.89</td>
<td>359.93</td>
<td>9.99%</td>
</tr>
<tr>
<td>HU 10M</td>
<td>536.66</td>
<td>496.94</td>
<td>7.40%</td>
</tr>
<tr>
<td>HU 100M</td>
<td>437.31</td>
<td>407.75</td>
<td>6.76%</td>
</tr>
<tr>
<td>HU 460M</td>
<td>307.69</td>
<td>288.89</td>
<td>6.11%</td>
</tr>
</tbody>
</table>

Table 5.5: Perplexities of test sets using 4-gram modified Kneser-Ney models, as well as adding a $\nu\phi$-LM with $|C| = 400$ classes.
Figure 5.6: Interpolation weights for English, Arabic, Croatian, and Hungarian corpora, using $|C| = 400$ classes in the class-based model.
class-based models and $n$-gram models.

The Markov-order weights for the respective $n$-gram, class, and decaying language models behave similarly as the training set size increases. With smaller training sets, the lower orders (e.g., unigrams and bigrams) of each of these language models receive more weight compared to the higher orders. This situation reverses for the larger training sets, with the higher orders having more weight. This is not surprising since there are fewer attestations of higher-order sequences than lower-order ones, however given enough occurrences the higher-order sequences are more reliable. Across languages we see a similar story: at a given training set size, models for languages having the lower perplexity tend to have more weight at the higher orders, and vice versa.

We also evaluated the $\nu\phi$-LM without the class-based model. Given that we have seen the weights of the full $\nu\phi$-LM in Figures 5.5 and 5.6, we already have a good idea what to expect: fairly stable improvements across languages and training set sizes with the exception of English, and a convergence of perplexities between the decaying language model and the $\nu\phi$-LM at around $10^9$ training tokens. Table 5.6 lists these perplexities. Accompanying these tables is Figure 5.7, which shows the relative weights of each model as training size increases for each language. Improvements in perplexity over the 4-gram modified Kneser-Ney correspond to an increased weight for the decaying language model. All of the results with and without the class-based model are significant ($p \ll 0.0001$), using the paired Wilcoxon signed-rank test.

In general we do see a stable improvement across languages and training set sizes, with the least improvements occurring for English. This is not particularly surprising,
<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>+DKLM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN 1M</td>
<td>251.39</td>
<td>242.46</td>
<td>3.55%</td>
</tr>
<tr>
<td>EN 10M</td>
<td>187.69</td>
<td>181.33</td>
<td>3.38%</td>
</tr>
<tr>
<td>EN 100M</td>
<td>126.96</td>
<td>123.28</td>
<td>2.90%</td>
</tr>
<tr>
<td>EN 1B</td>
<td>83.91</td>
<td>82.20</td>
<td>2.04%</td>
</tr>
<tr>
<td>AR 1M</td>
<td>831.35</td>
<td>776.95</td>
<td>6.54%</td>
</tr>
<tr>
<td>AR 10M</td>
<td>678.63</td>
<td>629.21</td>
<td>7.28%</td>
</tr>
<tr>
<td>AR 100M</td>
<td>410.08</td>
<td>382.90</td>
<td>6.62%</td>
</tr>
<tr>
<td>AR 375M</td>
<td>286.17</td>
<td>269.50</td>
<td>5.83%</td>
</tr>
<tr>
<td>HR 1M</td>
<td>566.25</td>
<td>529.11</td>
<td>6.56%</td>
</tr>
<tr>
<td>HR 10M</td>
<td>601.73</td>
<td>554.83</td>
<td>7.79%</td>
</tr>
<tr>
<td>HR 100M</td>
<td>432.82</td>
<td>399.39</td>
<td>7.73%</td>
</tr>
<tr>
<td>HR 550M</td>
<td>310.67</td>
<td>289.12</td>
<td>6.94%</td>
</tr>
<tr>
<td>HU 1M</td>
<td>399.89</td>
<td>384.59</td>
<td>3.83%</td>
</tr>
<tr>
<td>HU 10M</td>
<td>536.66</td>
<td>508.45</td>
<td>5.26%</td>
</tr>
<tr>
<td>HU 100M</td>
<td>437.31</td>
<td>410.14</td>
<td>6.21%</td>
</tr>
<tr>
<td>HU 460M</td>
<td>307.69</td>
<td>288.93</td>
<td>6.10%</td>
</tr>
</tbody>
</table>

Table 5.6: Perplexities of test sets using 4-gram modified Kneser-Ney models, as well as adding a decaying language model.
Figure 5.7: Interpolation weights for English, Arabic, Croatian, and Hungarian corpora, not using the class-based model.
given that English has a more fixed word order than the other languages, and decaying language models ameliorate the effects of word-order variation in collocations and other expressions. We would thus also not expect much improvement in textual domains that heavily employ boilerplate language, such as parliamentary proceedings and patent applications.

On a related note, we can also observe that perplexity using the smallest English training set is lower than perplexity for all other languages, even with half of a billion training set tokens. This illustrates the disparity in linguistic typology between English and the other three languages. Most research on language modeling focuses on evaluating English test sets. This can lead to a distortion in language modeling research.

There is also an interesting trend with the decay constants. As the training set size increases, the value of the decay constants for each of the Markov orders in the decaying language model tends to decrease. Figures 5.8 and 5.9 illustrate the decay constants for each of the languages, with Figure 5.8 corresponding to the experiments of using the full $\nu\phi$-LM, with $|C| = 400$ word classes, and Figure 5.9 corresponding to the experiments using only the modified Kneser-Ney model with the decaying language model. This decrease in the decay constant value as the training set size increases could be because the additional training data provides a smoother distribution, and because the greater contribution of decaying language models at the higher end of training set sizes involves longer-distance lexical relations.

Further evidence comes from comparing the values of the decay constants in the extended bigrams vs. extended trigrams. Naturally the former has more data than the
Figure 5.8: Decay constants for English, Arabic, Croatian, and Hungarian corpora, using $|C| = 400$ classes.
Figure 5.9: Decay constants for English, Arabic, Croatian, and Hungarian corpora, not using the class-based model.
latter, and thus exhibits a smoother distribution. The Hungarian extended trigram decay constant at \(10^6\) is unusual here, however a closer inspection reveals that the weight for extended trigrams is set so low that the decay constant here is irrelevant. The weight is set as low as it is due to the high degree of morphological richness and word order variation of Hungarian, necessitating large training sets.

5.4.3 Implementational Details

An important consideration in using a language model is whether or not it is feasible to use given time and space constraints. Training time for the \(\nu\phi\)-LM scales linearly with the number of sentences in the training set. The largest training sets for each of the languages took 2–5 hours to train using a fairly slow processor (2.4GHz) with a small L2 cache (2MB), using a single thread on a single machine. The model lends itself naturally to distributed training, and in fact earlier versions of the code supported training within the MapReduce framework, although this method is not being actively maintained. Probabilities in the model file are not explicitly normalized, which allows for easy online updating of the model if additional training data is added. Normalization of each of the components is simple and lightweight, involving two lookups to a hash table and one division operation.

The time needed to tune the small number of dense parameters in the model is constant with respect to the size of the training data or model. Rather, it is dependent on the size of the development set, since it involves repeated querying of this set.\(^4\)

\(^4\)And since the model is represented in memory as a hash table.
is an easily parallelizable procedure with little overhead, and we used 10 threads per experiment. We used fairly large development sets ranging from 10 to 30 million tokens in size (cf. Table 5.3), so tuning times took 1–8 hours depending on the language.

The software can query 100–350 million words per second, depending on the language and processor. This step is also parallelized, with the constraint that the order of the output sentences must be in the same order as the input sentences. This requirement exacts a small time penalty, but it ensures sequential consistency (Lamport, 1979), which may be necessary in certain applications.
<table>
<thead>
<tr>
<th>Corpus Segment</th>
<th>Tokens</th>
<th>Utterances</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN SWB Training Set</td>
<td>883K</td>
<td>72K</td>
</tr>
<tr>
<td>EN SWB Dev Set</td>
<td>18,601</td>
<td>1,303</td>
</tr>
<tr>
<td>EN SWB Test Set</td>
<td>5,970</td>
<td>386</td>
</tr>
</tbody>
</table>

Table 5.7: Switchboard transcription training, development, and test set sizes.

### 5.5 Speech Recognition Experiments

“Two monologues do not make a dialogue.”

–Jeff Daly

#### 5.5.1 Data

We used the Switchboard corpus v1r2 (Godfrey and Holliman, 1997) with the ISIP/MS98 word transcriptions\(^5\) and the JHU WS97 data splits. This is a corpus of telephone conversations between strangers, discussing a given topic. All conversations are in English, and most of the interlocutors are native English speakers from throughout the United States. The telephone discussions were recorded in 1990 using the standard 8kHz sampling rate with 8 bits per sample. Table 5.7 lists the sizes of the training, development, and test sets. The mean length of utterances for these sets were 12, 14, and 15 words respectively.

The transcriptions include disfluencies, hesitations, and filler words. Here are some examples of these from the development set:

\(^5\)http://www.isip.piconepress.com/projects/switchboard
Utterances like these are quite common throughout the corpus, and are transcribed as such in the test set as well. These have the effect of driving up word error rate.

The corpus also includes many single word utterances, including “YEAH”, “UM-HUM”, “UH-HUH”, “RIGHT”, “HM”, “OKAY”, “YES”, “OH”, “HUH”, “HUM”, among many others. Common two-word utterances include “OH YEAH”, “YOU KNOW”, “THAT’S RIGHT”, “OH REALLY”, “UM-HUM UM-HUM”, among others. For these utterances a simple low-order $n$-gram model will perform extremely well, better than most longer-distance and/or complex models since less probability is reserved in the former models for long-distance lexical relationships. Furthermore, these one and two word utterances are very common in the corpus, occupying almost a third of the utterances in the development set.

We rerank the 100-best hypotheses from the Kaldi speech recognition toolkit (Povey et al., 2011). This Free research-oriented large-vocabulary continuous speech recognition system (LVCSR) has been written in C++ with an emphasis on extensible design. We used the included Switchboard training recipe (s5) to stage tri5a, which represents a triphone based system with speaker adaptive training using feature-space maximum likelihood linear regression (fMLLR).
Table 5.8: Perplexities of the Switchboard transcription test set using 4-gram modified Kneser-Ney models, as well as adding a $\nu\phi$-LM with $|C| = 200$ classes (top), or a decaying model (bottom). 

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\nu\phi$-LM PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN SWB 10K</td>
<td>100.22</td>
<td>91.69</td>
<td>8.51%</td>
</tr>
<tr>
<td>EN SWB 100K</td>
<td>115.11</td>
<td>102.75</td>
<td>10.74%</td>
</tr>
<tr>
<td>EN SWB 883K</td>
<td>98.20</td>
<td>90.52</td>
<td>7.82%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MKN4 PP</th>
<th>$+\text{DKLM}$ PP</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN SWB 10K</td>
<td>100.22</td>
<td>95.86</td>
<td>4.35%</td>
</tr>
<tr>
<td>EN SWB 100K</td>
<td>115.11</td>
<td>111.38</td>
<td>3.24%</td>
</tr>
<tr>
<td>EN SWB 883K</td>
<td>98.20</td>
<td>96.28</td>
<td>1.96%</td>
</tr>
</tbody>
</table>

5.5.2 Results and Analysis

For reference, we first present perplexity results of this dataset. We used the same experimental procedures as in Section 5.4. Because the corpus is considerably smaller than the other textual corpora, we start the training set size at $10^4 = 10K$ words, incrementing the exponent until reaching the full training set of 883K words. For the $\nu\phi$-LM we used $|C| = 200$ classes due to the small size of the corpus. The results are presented in Table 5.8.

We see a considerable reduction in perplexity by adding the $\nu\phi$-LM. All of the results with and without the class-based model are significant ($p < 0.05$), using the paired Wilcoxon signed-rank test. At $10^4$ training tokens, almost half of the gains are attributable to the decaying language model and the other half to the class-based
<table>
<thead>
<tr>
<th>Corpus</th>
<th>3gram WER</th>
<th>+νφ-LM WER</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN SWB 883K</td>
<td>35.99</td>
<td>35.96</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 5.9: Word error rates of the Switchboard development set using trigram models, as well as adding a νφ-LM with |C| = 200 classes (top), or a decaying model (bottom).

model, as is also evidenced in Figure 5.10. With the addition of more training tokens, eventually the class-based model takes more of the interpolation weight, and the weight for the decaying language model stabilizes. Without the class-based model the weights for the decaying language model are about the same at 10^4 training tokens, and they almost double with a larger training set over their equivalents in the full νφ-LM. The remaining balance of the weights are reallocated to the n-gram model.

Next we look at the word error rate results. The development set and test set results are presented in Tables 5.9 and 5.10 respectively, with the νφ-LM and decaying language model showing negligible difference. The development set oracle word error rate—the lowest error rate possible given the 100-best list produced by Kaldi—is 25.56%, and the test set oracle word error rate is 24.04%. In light of the lower word error rates discussed in Section 5.3.2 at higher perplexities, the Switchboard corpus presents a more challenging task given that we are working with perplexities in the 90–115 range.

Using only the acoustic model in the development set 100-best list yields a word
Figure 5.10: Interpolation weights for English Switchboard corpus 1r2, with the first plot using $|C| = 200$ classes. The second plot is not using a class-based model.
<table>
<thead>
<tr>
<th>Corpus</th>
<th>3gram WER</th>
<th>+$\nu$-LM WER</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN SWB 883K</td>
<td>32.85</td>
<td>32.86</td>
<td>--</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>3gram WER</th>
<th>+DKLM WER</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN SWB 883K</td>
<td>32.85</td>
<td>32.81</td>
<td>--</td>
</tr>
</tbody>
</table>

Table 5.10: Word error rates of the Switchboard test/eval set using trigram models, as well as adding a $\nu$-LM with $|C| = 200$ classes (top), or a decaying model (bottom).

error rate of $38.59\%$, while using only the acoustic model in the lattice gives a word error rate of $52.31\%$. These figures show that this is a fairly good acoustic model. Excluding the acoustic model in the 100-best list gives a word error rate of $40.01\%$, and excluding the acoustic model from the lattice gives a word error rate of $43.81\%$.

It is important to note that Kaldi mixes language model scores with transition model scores and pronunciation probabilities into a decoding graph, impeding controlled experimentation on the language model itself. There is a way of subtracting the language model scores from the decoding graph, but only on an entire lattice. The process of subtracting the language model from the decoding graph is described in the documentation\(^6\) as thus:

Suppose that we are given an LM-scale $S$ and a LM $\text{G.fst}$. We first multiply all the costs in $\text{G.fst}$ by $S$. Then for each lattice, we compose it on the right with $G$, lattice-determinize it (to keep only the best path through $G$ for each word sequence) and write it out. This would work fine for positive $S$, but for negative $S$ it would be equivalent to taking the worst path through $G$, for each word sequence. To fix this, we instead do it as follows. For each input lattice,
we first scale the lattice’s graph (or LM) costs by the inverse of \( S \); we then compose on the right with \( G.fst \); we run lattice-determinization [...] on the resulting lattice which retains only the best path through the lattice for each word sequence; and we then scale the lattice’s graph/LM scores by \( S \). This does the right thing for negative \( S \). This whole approach only makes sense if the input lattice has only a single path through it for each word sequence (e.g. if it was lattice determinized). We assume that all lattices passed between programs have this property (this is why it is not a good idea to write out “raw” state-level lattices, unless you know what you are doing).

Note that in order for the composition to work, this program needs to map \( G.fst \) from the tropical semiring into the LatticeWeight semiring. It does this by putting the weights of \( G.fst \) into the first part of the weight (the “graph” part), and leaving the second part of the weight zero (in the semiring, this is 1). At the C++ level, this mapping is done using the MapFst mechanism of OpenFst, in which we define a Mapper class to map StdArc to LatticeArc, and then create an object of type MapFst which is evaluated on demand and converts \( G.fst \) to the LatticeWeight weight type.

The effect of this is a lattice that gives slightly better results than an acoustic model-only lattice, but is not useful for language modeling reranking experiments, since we still end up with an poor \( n \)-best list that is dominated by high-ranking acoustic model scores. The word error rate for an acoustic model-only lattice for this corpus is 52.31%, while the word error rate for a language model-subtracted lattice (having thus acoustic model scores, transition probabilities, and pronunciation probabilities) is 52.29%. There is a decrease in insertions from the former lattice to the latter (2569 vs. 2504), but an increase in deletions (731 vs. 758) and substitutions (6431 vs. 6464).

If we rerank the 100 best hypotheses of either of these two lattices using either a 4-gram modified Kneser-Ney model or a \( \nu \phi \)-LM, there is a 7–9% improvement in word
error rate, but the original 100-best list for either lattice has oracle word error rates in the 41.4–41.6% range. Hence using a better-quality language model on either of these $n$-best lists cannot overcome the inherent limitations of using an $n$-best list generated by the language model-subtracted lattice (or the acoustic model-only lattice). In addition to these inherent limitations for practical speech recognition usage, there are limitations and caveats associated with conducting reranking experiments comparing a 4-gram modified Kneser-Ney model with the addition of a $\nu\phi$-LM. If the interpolation weights and other parameters of the $\nu\phi$-LM are tuned on perplexity, then they could produce worse results than not using this model. On the other hand, if these weights and parameters are tuned on word error rate for either of these lattices, then they would be optimized on distorted data that does not represent a normal usage scenario.

Let us now turn to a better lattice, in which the interpolation weights of acoustic model and decoding graph has been optimized on word error rate. This will produce a better $n$-best list with which to work. The optimal ratio here of acoustic model to decoding graph scores for the Switchboard corpus is approximately 0.0714 : 1 . Note that neither one of these scores is normalized, so this ratio does not necessarily mean that the decoding graph score has more influence than the acoustic model score in the decoding process.

After the $n$-best list is generated, we can nullify various scores to get an idea of the word error rate ranges we are dealing with. If the acoustic model scores are set to zero at this point, the word error rate goes from 35.99% in the optimal setting to 40.01% . Conversely, if the decoding graph scores are set to zero, the word error rate is 38.59% . Hence the acoustic model appears to have more of an effect on word error
rates than the decoding graph does in this setting. Interestingly, the reverse of this is true in dealing with lattices. That is, the best word error rate is substantially higher when the decoding graph is not used in the lattice than when the acoustic model is not used (52.31% vs. 43.81% respectively). This further confirms that using language models to rerank n-best lists for this task, rather than reranking lattices, is of very limited utility, in spite of the fact that the oracle word error rate is 25.56%.

If we add a 4-gram modified Kneser-Ney model to the 100-best list that does not use the decoding graph scores (and hence the built-in language model scores), the word error rate drops only slightly, from 38.59% to 38.36%—additional confirmation of the limited utility of language models here. Using exclusively the $\nu\phi$-LM gives a word error rate of 38.55%, and using both external language models gives 38.51%. In light of all these facts it is not surprising that adding any of the external language models to a 100-best list that uses both an acoustic model and a decoding graph makes almost no difference: from 35.99% to 35.96% for either the 4-gram modified Kneser-Ney, $\nu\phi$-LM, or MKN4+$\nu\phi$-LM. The results of all these different configurations are summarized in Table 5.11.

If we assume for a moment that this corpus and setup are similar to the Eval96 setup in Klakow and Peters (2002) (cf. § 5.3.2)—this may or may not be the case—then we might extrapolate what kind of error rates we should expect with extremely good perplexities. Both setups follow similar trajectories, but at non-overlapping $x$ ranges. Using the Eval96 function we would expect a word error rate for our system of 35.77%, since our development set perplexity is a respectable 97.37. This error rate is fairly

---

7 The language model interpolation weights were fixed, tuned on perplexity.
<table>
<thead>
<tr>
<th>Lattice Configuration</th>
<th>Lattice WER</th>
<th>100-best Configuration</th>
<th>100-best WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWB Dev AM=0 G=1</td>
<td>43.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWB Dev AM=1 G=0</td>
<td>52.31</td>
<td>AM=1 G=0 MKN4=best</td>
<td>47.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=0 $\nu\phi$-LM=best</td>
<td>48.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=0 MKN4+$\nu\phi$-LM=best</td>
<td>47.57</td>
</tr>
<tr>
<td>SWB Dev AM=1 LM=0 (subtracted)</td>
<td>52.29</td>
<td>AM=1 G=1 MKN4=best</td>
<td>47.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=1 $\nu\phi$-LM=best</td>
<td>48.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=1 MKN4+$\nu\phi$-LM=best</td>
<td>47.48</td>
</tr>
<tr>
<td>SWB Dev AM=0.0714 G=1</td>
<td>35.99</td>
<td>AM=0 G=1</td>
<td>40.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=0</td>
<td>38.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=0 MKN4=best</td>
<td>38.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=0 $\nu\phi$-LM=best</td>
<td>38.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=0 MKN4+$\nu\phi$-LM=best</td>
<td>38.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=1 MKN4=best</td>
<td>35.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=1 $\nu\phi$-LM=best</td>
<td>35.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AM=1 G=1 MKN4+$\nu\phi$-LM=best</td>
<td>35.96</td>
</tr>
</tbody>
</table>

Table 5.11: Word error rate percentages of the Switchboard development set using various lattices and 100-best lists. Here $AM$ refers to the acoustic model cost interpolation weight and $G$ refers to the decoding graph cost interpolation weight, which graph is the product of the language model cost, transition probabilities, and any pronunciation cost. The interpolation weights here need not sum to one.
close to the actual 35.96%. A language model getting a perplexity of 60 on the development set, which is extremely good, would produce a word error rate of only 32.9%.
Chapter 6

Conclusion

We have shown that experimental results from the neuroscience community can inspire statistical models of language that have substantive perplexity improvements over commonly-used existing models. As the speech recognition experiments were inconclusive, we would like to use existing lattices used in other papers for further comparisons. We are also in the process of evaluating our model with RNNLM (§3.5.2), which exhibits some structural similarities to the decaying language model.

There has been a somewhat implicit assumption in this work of a relationship between ERP magnitude and perplexity, but this connection could be evaluated explicitly.\(^1\) Furthermore, we could use current knowledge of the temporal processing of sentences (cf. Friederici, 2002) to dynamically adapt parameters, with the early components possibly influencing later components.

6.1 Future Work

There are several future directions to take this work, which are either not within the scope our expertise, are tangential to the current line of research, require several

\(^1\)Related to this is the interesting idea of using EEG signals to directly improve speech recognition results (Chen et al., 2012).
months of implementation, or are in under-explored areas.

### 6.1.1 Local Lexical Processing

While there has been much work among those working on language-related ERPs on both local and long-distance syntactic processing, as well as long-distance lexical-semantic processing, little attention has been paid to local lexical processing. This would involve idiomatic expressions, multiword proper nouns, and titles of popular films & songs, for example. Initially it might seem as though these would be processed in the same way as what is involved in the N400. But there are differences.

Firstly, the N400 is cumulative with the entire sentence (Kutas and Hillyard, 1988), while the type of expressions mentioned above would be cumulative only within their own local phrases. Secondly, function words play a minimal role in the N400, but such words would have equal contribution in the prediction of a local lexical expression. For example, in the expression “he went up the . . .”, there are no content words involved\(^2\), but they do play an important role in expecting a subsequent word like hill, mountain, or creek. Thus it is unclear without experimental evidence whether violations of local lexical expressions like “that takes the pie” would elicit an N400, an ELAN, or something else entirely.

The N280/LPN does exhibit some properties that would be in line with these expectations. It occurs for both open-class and closed-class words (Neville et al., 1992), and in fact it is slightly more sensitive to closed-class words (Garnsey, 1985; Van Petten and Kutas, 1991; Brown et al., 1999). ter Keurs et al. (1999) showed

\(^2\)Or minimally with the semantically-bereft light verb go.
that the N280/LPN related the processing of open-class vs. closed-class words with syntactic integration, which contrasts with the relatively syntax-independent N400. But this ERP component is less studied than the N400 or ELAN, and most of the research has focused on the related concepts of lexical frequency, the open-class vs. closed class distinction, and word length (cf. Osterhout et al., 2002).

6.1.2 Model

There are a number of future directions for potentially improving the decaying model. It is currently unpruned, which results in larger models than are necessary. Low-frequency words could be ignored, with the allocated $P_d$ from Equation 4.11 being redistributed elsewhere (or to the unknown $\textlt{unk}\textrt$ token). This frequency threshold can be determined beforehand, resulting in a more principled pruning, or alternatively could be set once a maximum amount of memory has been reached.

Another area to explore in the future is a hybrid decaying–n-gram language model. This could involve training an n-gram model using modified Kneser-Ney smoothing, then using this distribution instead of $P_d$ within the larger decaying model in Equation 4.12. The distribution $P_d$ is already a generalization of n-gram models, specifically Jelinek-Mercer models. Using modified Kneser-Ney smoothing might provide additional improvements due to it accounting for history diversity.
6.1.3 Adaptation to Machine Translation

The factored phrase-based Moses machine translation system\(^3\) (Koehn et al., 2007) allows for flexible integration of multiple language model components, and seamless LM interpolation with the tuning of other parameters. It is a mature project, reasonably fast, and widely deployed.

We would like to evaluate our approach using German-English and Urdu-English parallel corpora. The German-English data comes from the shared task of the 2011 Workshop on Statistical Machine Translation (WMT-2011: Callison-Burch et al., 2011).\(^4\) This consists of approximately 50 million tokens per language, comprising 1.8 million sentence pairs. The Urdu-English data comes from the Nist OpenMT-2008 Evaluation.\(^5\), which consists of approximately 1.6 million words per language, comprising 87.7 thousand sentence pairs. It also includes an additional 114 thousand word or phrase pairs. The test set for this language pair comes from the OpenMT-2009 dev set. Table 6.1 summarizes the training corpora used for our experiments. We also

\(^3\)http://www.statmt.org/moses
\(^4\)http://www.statmt.org/wmt11/translation-task.html
\(^5\)http://www.itl.nist.gov/iad/mig/tests/mt/2008
would add the English Gigaword corpus (Parker et al., 2011b) for building English language models—both the word $n$-gram language model and our model.

### 6.1.4 Adaptation to Software Keyboards

In addition to the well-understood tasks of machine translation and speech recognition, a statistical language model can also be used to provide better accuracy for software keyboards in tablets and smartphones. Language models can be used here in much the same way as they are used in speech recognition—as a prior. While the most basic software keyboard might choose a key based on the location of the centroid of the area pressed by a finger, a slightly more intelligent virtual keyboard would subsequently spellcheck a completed word and autocorrect if necessary. However, we can view touchscreen software keyboards as functioning similarly to speech recognition or optical character recognition, where in this context $P(Y|X)$ would be a spatial model, and $P(X)$ a language model.

Firstly, we would split a large text corpus, such as the Enron email corpus, into the usual training, development, and test sets. Then we would distort the development and test sets such that the probability of realizing a given letter is based on a two-
dimensional Gaussian distribution from the centroid of the spatial placement of that letter on the virtual keyboard. The variance would be increased at different intervals, roughly corresponding to an increase in fat-fingering. This application could have widespread benefits with the widespread use of smartphones and tablets.

6.1.5 Implementation

Finally, there are several implementational improvements worth pursuing. The software could use memory mapping, such as `mmap`, to allow for much larger models to be used. Memory mapping treats hard drives and physical memory in a uniform manner, allowing disk pages containing sections of the model to be loaded into physical memory on-demand. KenLM uses this approach, and has demonstrated its scalability (Heafield et al., 2013). Using memory mapping, rather than allocating memory in the traditional way (`malloc`), would require substantial code refactoring. Furthermore, `mmap` is not defined in the C standard, but rather is a Posix system call, which would restrict the code to Unix-like operating systems. Windows systems would need a different system call, namely `MapViewOfFile`. Alternatively, we could support more compact data structures like tries (de la Briandais, 1959; Fredkin, 1960).

Due to the evolution of the model and it implementation, the software is restricted to using extended bigrams and extended trigrams in the decaying model. Higher-order extended \( n \)-grams are not as necessary in a decaying language model as higher-order \( n \)-grams are in a traditional \( n \)-gram model, however it would be nice to generalize the code to support arbitrary order extended \( n \)-grams. Doing so would require moderate
code refactoring, with probably modest gains in perplexity.
References


Dugast, Christian, Reinhard Kneser, Xavier Aubert, Stefan Ortmanns, Klaus Beulen, and Hermann Ney. 1995. Continuous speech recognition tests and results for the NAB’94 corpus. In Proceedings of the ARPA Spoken Language Technology Workshop, 156–161, Austin, TX, USA.


Garofalo, John, David Graff, Doug Paul, and David Pallett, 1993. CSR-I (WSJ0) Sennheiser. Linguistics Data Consortium, University of Pennsylvania. LDC Catalog No.: LDC93S6B.

Garofalo, John, David Graff, Doug Paul, and David Pallett, 1994. CSR-II (WSJ1) Sennheiser. Linguistics Data Consortium, University of Pennsylvania. LDC Catalog No.: LDC94S13B.


OpenMP ARB, 2013. OpenMP Application Program Interface.


131


Xu, Wei, and Alex Rudnicky. 2000. Can Artificial Neural Networks Learn Language Models? In *Sixth International Conference on Spoken Language Processing*. 133
Appendix A

Neural Activity and its Measurement

“I can’t give you a brain, but I can give you a diploma.”

–The Wizard of Oz

In order to be able to quantify how the brain processes language-related activities, we must first look at how the neurons in the brain work, and how various instruments measure neural activity, either directly or indirectly.

Animal cells maintain a trans-membrane potential, or electric field inside the membrane, which is approximately -65 mV to -70 mV at equilibrium. Voltage-gated ion channels in the plasma membrane control the flow of sodium ions, which are crucial to the function of neurons. As sodium ion channels allow positively-charged sodium ions in, the potassium ion channels allow potassium ions to flow out, which usually results in a return to the resting membrane potential (Nuñez and Srinivasan, 2006). Another method for maintaining this equilibrium in the other direction is the Na⁺/K⁺-ATPase, an enzyme that pumps three sodium ions out for every two potassium ions in (Hall and Guyton, 2006).

However, if more sodium ions flow in than potassium ions flow out, the voltage may exceed a necessary threshold of 15 mV higher than the resting potential. When
this occurs, a positive feedback loop occurs, as increases in the membrane potential cause more ion channels to open, which increases the membrane potential. It is this remarkable feedback loop that gives neurons (inter alia) their nonlinear conductive properties, which prevents degradation over long distances (Nuñez and Srinivasan, 2006). As a result of the increase of cations within the membrane, the voltage increases quickly until it reaches the sodium ion equilibrium voltage of about $+55\text{mV}$, where the sodium ion channels are completely open (Bullock et al., 1977).

The membrane permeability to sodium ions does not last long, as the membrane at this stage becomes permeable to potassium ions, and the sodium channels become inactivated (Purves et al., 2008). The increase in potassium ion permeability in the
membrane causes repolarization. This repolarization is quite rapid, and results in an undershoot of the membrane potential relative to the resting potential. This phase is referred to as afterhyperpolarization. As $K^+$ permeability subsides, the voltage levels eventually return to that of the resting membrane potential (Purves et al., 2008). This entire process is illustrated in Figure A.1 for the action potential of rat pyramidal neurons at the hippocampus.

Neurons in the brain do not work in isolation—an action potential by its very nature involves multiple neurons. At a larger scale, certain regions of the brain have shown patterns for specialization of specific functions.\(^1\) For example, the inferior frontal gyrus (BA 44 & 45) is involved with local sequential processing, while the middle temporal gyrus, shown in Figure A.2, plays a role in processing lexical semantics.

\(^1\)Although there is some flexibility in this regard, especially among those who have had strokes or other brain lesions.
(Friederici et al., 2000; Poeppel et al., 2008). Taken at a large scale, action potentials emit changes in voltage across the neocortex which are both temporally- and spatially-measurable. These changes allow us to understand how, when, and where natural language is processed.

A.1 Electroencephalography

Electroencephalograms (EEG) record voltage fluctuations across the cerebral cortex (Nuñez and Srinivasan, 2006), which assumes population coding (cf. Wu et al., 2002). They do so using many electrodes placed on the scalp in an array. The locations of these electrodes are based on the international 10–20 system (cf. Jasper, 1958), and are placed over the frontal, temporal, parietal, and occipital lobes of both hemispheres, as well as over the center of the scalp.

Electroencephalography offers many advantages in measuring neural activity, relative to other techniques. Compared to behavioral experiments, EEG is both continuous and online. Thus we can measure how the brain responds to various stimuli as they are unfolding, giving much more information to base scientific interpretations upon.

Compared to the neuroimaging techniques discussed in section A.2, EEG is a direct measure of neural activity, and provides orders of magnitude higher temporal resolution. This is very important to understanding the temporal progression of processing or generating natural language. Electroencephalography is also less invasive
than positron emission tomography or near-infrared spectroscopy, and much less expensive than any neuroimaging technique including magnetoencephalography.

A.2 Hemodynamic Neuroimaging Techniques

Neuroimaging techniques such as functional magnetic resonance imaging (fMRI), positron emission tomography (PET), and near-infrared spectroscopy (NIRS) rely on changes in blood flow in the brain to measure activations in the brain to infer localized neural activity. This inherently restricts temporal resolution, since there is a lag between neural activity and additional blood supply. These techniques currently have temporal resolution in terms of seconds. They are also generally quite expensive relative to electroencephalography. Their main strength over EEG lies in spatial resolution in terms of millimeters.
Appendix B

Association Measures

“You shall know a word by the company it keeps.”

– J. R. Firth

Association measures quantify the degree to which two variables are related.\textsuperscript{1} In the context of this work, we apply such measures to words. The relation between a word \( w_1 \) and another word \( w_2 \) are often not symmetric in practice, although this can be decided upon by what one considers to be a co-occurrence of \( w_1 \) and \( w_2 \). That is, a symmetric joint distribution \( P(w_1, w_2) \) would be the sum of an asymmetric \( P(w_1, w_2) \) and \( P(w_2, w_1) \).

In information retrieval contexts, where the ordering of words within a document is often irrelevant, words may be counted as co-occurring if they are in the same document, regardless of whether \( w_1 \prec w_2 \) or \( w_2 \prec w_1 \). In other contexts, however, the linear precedence relation may be relevant. Incremental language modeling is such a case—the joint distribution of \( P(\text{the, book}) \) will differ considerably from \( P(\text{book, the}) \).

In Table B.1 we introduce labels for various joint and marginal distributions involving a word \( w_1 \) and another word \( w_2 \). The formulation of this table and Equation \textsuperscript{1}Church and Hanks (1990) give an excellent introduction to the application of one association measure, pointwise mutual information (PMI), to lexicography.
Table B.1: Labels for various joint and marginal distributions involving $w_1$ and $w_2$. Here $\neg w_1$ denotes $\{w|w \in V, w \neq w_1\}$, or the complement of $w_1$.

B.1 are derived from documentation in the CPAN module Text::NSP (Banerjee and Pedersen, 2003).

From four of the distributions in Table B.1 we can derive the others, saving considerable computational and storage costs. We use the counts $C$ of $w_1$, $w_2$, their
co-occurrence $w_1, w_2$, and total number of words in a training set:

\[
\begin{align*}
n_{11} &= C(w_1, w_2) \\
n_{1p} &= C(w_1) \\
np1 &= C(w_2) \\
npp &= \sum_{w \in V} C(w) \\
np2 &= npp - np1 \\
n12 &= n1p - n11 \\
n21 &= np1 - n11 \\
n22 &= np2 - n12 \\
m_{11} &= \frac{np1 \cdot n1p}{npp} \\
m_{12} &= \frac{np2 \cdot n1p}{npp} \\
m_{21} &= \frac{np1 \cdot n2p}{npp} \\
m_{22} &= \frac{np2 \cdot n2p}{npp}
\end{align*}
\]

where $m_{21}$, for example, is the expected value of $n_{21}$. From these definitions we can list the values that are required for each association measure, in Table B.2.
Table B.2: Various association measures and the joint and/or marginal distributions that they use.
Author Index

Abramson (1963), 21, 113
Anderson and Lebiere (1998), 3, 113
Bahl et al. (1983), 69, 113
Baker (1979), 39, 63, 113
Balota et al. (1985), 48, 113
Banerjee and Pedersen (2003), 113, 140
Bean (2007), xi, 113, 135
Bengio and Bengio (2000a), 27, 113
Bengio and Bengio (2000b), 27, 114
Bengio et al. (2000), 27, 114
Bengio et al. (2003), ix, xii, 1, 27, 28, 30, 33, 114
Bengio (2002), 41, 113
Beyerlein et al. (1998), 73, 114
Bodén (2002), ix, 31, 32, 114
Brants et al. (2007), ix, 2, 15, 16, 66, 114
Brants et al. (2007), 17
Brown et al. (1979), 6, 115
Brown et al. (1992a), 18, 37, 43, 114
Brown et al. (1992b), 19, 114
Brown et al. (1999), 107, 114
Brun et al. (2007), 20, 115
Bullock et al. (1977), 115, 135
Buntine (1994), ix, 23, 115
Callison-Burch et al. (2011), 109, 115
Carroll and Slowiaczek (1986), 49, 115
Cassimatis (2002), 3, 115
Chelba and Jelinek (1998), 2, 39, 115
Chelba et al. (2014), 35, 115
Chelba (2000), 2, 39, 63, 115
Chen and Goodman (1998), ii, ix, 14, 16, 29, 34, 36, 37, 56, 116
Chen et al. (2009), 37, 116
Chen et al. (2012), 106, 116
Chen (1995), 63, 116
Chen (2009), 37, 116
Chong et al. (2013), 20, 116
Church and Hanks (1990), 116, 139
Clark (2003), 44, 68, 116
Cormen et al. (2001), 15, 116
Darroch and Ratcliff (1972), 36, 117
Dean and Ghemawat (2004), 2, 117
Dean and Kanazawa (1989), ix, 24, 117
Deerwester et al. (1990), 23, 26, 117
Dempster et al. (1977), 23, 117
Derouault and Mérialdo (1984), 18, 117
Derouault and Mérialdo (1986), 18, 117
Dugast et al. (1995), 72, 117
Dunning (1993), 46, 118
Dyer et al. (2010), 68, 118
Ehrlich and Rayner (1981), 48, 118
Elman (1990), ix, 30–33, 118
Emami et al. (2003), 1, 118
Ferreira and Henderson (1990), 3, 118
Fiebach et al. (2001), 10, 118
Fiebach et al. (2002), 10, 118
Fischler et al. (1983), 6, 118
Fodor and Frazier (1980), 3, 119
Frazier (1979), 3, 119
Frazier (1987), 3, 119
Fredkin (1960), 111, 119
Friederici and Frisch (2000), 8, 9, 119
Friederici et al. (1993), 6, 119
Friederici et al. (1996), 6, 8, 119
Friederici et al. (1998), 9, 119
Friederici et al. (1999), 6, 119
Friederici et al. (2000), 119, 137
Friederici (2002), 6, 106, 119
Frisch et al. (2002), 9, 120
Frisch et al. (2004), 6, 120
Gamba et al. (1961), 26, 120
Gao and Suzuki (2003), 63, 120
Garnsey (1985), 107, 120
Garofalo et al. (1993), 34, 120
Garofalo et al. (1994), 34, 120
Gibson (1991), 10, 120
Gibson (1998), 10, 120
Gildea and Hofmann (1999), ix, 2, 23, 24, 120
Gobet and Lane (2010), 3, 120
Godfrey and Holliman (1997), 56, 73, 95, 121
Goodman (2000), 19, 20, 121
Goodman (2001), x, 19, 20, 25, 74–76, 121
Graff and Cieri (2003), 34, 121
Graff (1995), 72, 121
Gray (1918), xi, 121, 136
Guthrie et al. (2006), 20, 121
Hain et al. (2006), 35, 121
Halászy et al. (2004), 79, 121
Hall and Guyton (2006), 122, 134
Hasson and Small (2008), 1, 122
Heafield et al. (2013), 68, 111, 122
Heafield (2011), 68, 122
Hess et al. (1995), 48, 122
Hofmann (1999), 23, 122
Huang et al. (1993), 19, 122
Jasper (1958), 122, 137
Jaynes (1957), 36, 122
Jelinek and Lafferty (1991), 42, 122
Jelinek and Mercer (1980), x, 14, 36, 40, 55, 123
Jelinek et al. (1991), 19, 123
Jordan (1986), 30, 123
Just and Varma (2007), 3, 123
Kaan et al. (2000), 9, 11, 123
Katz (1987), 14, 35, 123
Kernighan and Ritchie (1988), 68, 123
Kieras and Meyer (1997), 3, 123
King and Kutas (1995), 11, 123
Klakow and Peters (2002), x, xii, 71–76, 103, 124
Klakow (1998), 25, 40, 123
Klein and Manning (2004), 65, 124
Knese and Ney (1993), 19, 44, 68, 124
Knese and Ney (1995), 27, 66, 124
Knese and Steinbiss (1993), 40, 124
Koehn et al. (2007), 68, 109, 124
Koehn (2010), 2, 124
Kokinov (1994), 3, 124
Kučera and Francis (1967), 22, 28, 125
Kuhn and De Mori (1990), 18, 124
Kuperberg et al. (2003), 6, 125
Kutas and Hillyard (1980), 6, 125
Kutas and Hillyard (1983), 7, 47, 125
Kutas and Hillyard (1988), 6, 107, 125
Kutas et al. (2006), 11, 125
Kutas et al. (2012), 5, 125
Lamport (1979), 94, 125
Langley et al. (2005), 3, 125
Lau et al. (1993a), 21, 37, 125
Lau et al. (1993b), 21, 35–37, 126
Levenshtein (1966), 70, 126
Lewis (2009), 2, 126
Liu and Nocedal (1989), 37, 126
Ljubešić and Erjavec (2011), 79, 126
Macdonald et al. (1994), 3, 126
Martin et al. (1998), 44, 68, 126
McKinnon and Osterhout (1996), 8, 126
McRae et al. (1998), 3, 126
Mecklinger et al. (1995), 9, 126
Miikkulainen and Dyer (1991), 26, 27, 127
Mikolov et al. (2010), 27, 31, 33, 127
Mikolov et al. (2011), 35, 127
Mnih and Teh (2012), 35, 127
Neely (1977), 49, 127
Neville et al. (1991), 6–8, 127
Neville et al. (1992), 107, 127
Newell (1990), 3, 127
Nuñez and Srinivasan (2006), 127, 134, 135, 137
Och (1999), 68, 127
Osterhout and Holcomb (1992), 8, 9, 128
Osterhout and Mobley (1995), 8, 128
Osterhout et al. (2002), 108, 128
Papineni et al. (2002), 76, 78, 128
Parker et al. (2009), 60, 79, 128
Parker et al. (2011a), 79, 128
Parker et al. (2011b), 2, 110, 128
Parker (1985), 26, 27, 31, 128
Paul and Baker (1992), 37, 128
Phillips et al. (2005), 11, 129
Pickhardt et al. (2014), 20, 129
Poeppe et al. (2008), 129, 137
Posner and Snyder (1975), 48, 129
Post et al. (2013), 68, 129
Povey et al. (2011), 96, 129
Purves et al. (2008), 129, 135, 136
Roark (2001), 43, 63, 129
Roark (2002), 43, 63, 129
Rosenblatt (1957), 26, 130
Rosenfeld and Huang (1992), 21, 22, 130
Rosenfeld et al. (2001), 39, 130
Rosenfeld (1994), 20, 21, 35, 130
Rosenfeld (1996), 21, 22, 130
Rumelhart et al. (1985), 31, 130
Rumelhart et al. (1986), 31, 130
Russell and Norvig (2003), 26, 130
Schneider and Shiffrin (1977), 48, 130
Schwartz et al. (2011), 63, 130
Schwenk and Gauvain (2002), 30, 131
Schwenk et al. (2012), 30, 131
Schwenk (2007), 1, 30, 130
Schwenk (2010), 1, 30, 131
Simpson et al. (1989), 48, 131
Snaider et al. (2011), 3, 131
Stanovich and West (1983), 48, 131
Steinhauer and Connolly (2008), 6, 8, 131
Steinhauer et al. (1999), 9, 131
Stemmer and Whitaker (1998), 5, 131
Stern (1996), 72, 74, 131
Stolcke (1995), 43, 132
Storm (1987), xi, 132, 135
Sun (2006), 3, 132
Tan et al. (2011), 2, 26, 132
Tan et al. (2012), 2, 26, 132
Trueswell et al. (1993), 3, 132
Tweedy et al. (1977), 49, 132
Van Petten and Kutas (1990), 7, 46, 133
Van Petten and Kutas (1991), 7, 47, 107, 133
Van Petten (1993), 7, 132

145
Wang et al. (2005), 2, 133
Wang et al. (2006), 2, 133
Werbos (1974), 26, 27, 31, 133
Werbos (1988), 31, 133
Werbos (1990), 31, 133
Widrow (1962), 26, 133
Wu et al. (2002), 133, 137
Xu and Rudnicky (2000), 1, 26, 133

beim Graben et al. (2008), 9, 113
de la Briandais (1959), 111, 117
ter Keurs et al. (1999), 107, 132
ISO/IEC JTC1/SC22/WG14 (1999), 68, 122
OpenMP ARB (2013), 68, 128
den Heyer et al. (1983), 49, 117
den Heyer (1985), 49, 117
Index

χ² measure, 142
10–20 system, 137

acoustic model, 34, 37, 39, 98, 100–104
action potential, 135, 136
afterhyperpolarization, 136
AP news corpus, 28–30, 79
Arabic, ii, v, vi, 15–17, 60, 66, 79–81, 84, 86, 89, 91, 92
script, 81
ASR, see speech recognition
association measure, 47, 49, 139, 141, 142
autocorrect, 110
average mutual information, 21, 142
backoff smoothing, 14, 35
backpropagation, 26, 27, 31
through time, 31, 33
bag-of-words, 25
basketball, 18
Beckett, Samuel, 44
bigram, 19, 22, 25, 27, 35, 36, 50–53, 55, 78, 87
Bleu, 15–17, 66, 76
Küntelichten, 109
boilerplate, 90
brain, ii, 1, 3, 134, 136–138
brevity penalty, 78
Brodmann area, 136
Brown algorithm, 44
Brown clustering, 37
Brown corpus, 22, 28–30
C, 68, 111
cache language model, 18, 19, 21, 22, 38, 71
cation, 135
cdec, 68
centroid, 110, 111
class-based language model, 18, 38, 43, 44, 71, 82, 84, 86, 87, 89, 92, 97–99
CNV, 6
coarticulation, 39
conditional probability, 142
CPAN, 140
CPS, 6
creek, 107
Croatian, ii, 79, 81, 84, 86, 89, 91, 92
cross-entropy, 69
DBN, see dynamic Bayesian network
decay, 53, 57
constant, 47, 49, 50, 54, 90–92
language model, 52, 52, 53, 55–57, 62, 81, 82, 87, 88, 90, 97, 98, 100, 106, 108, 111
deduplication, 80
diacritic, 81
Dice’s coefficient, 49, 142
Dijkstra, Edsger, 68
diploma, 134
disfluency, 55, 57, 67, 95
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>45, 96</td>
<td></td>
</tr>
<tr>
<td>dynamic Bayesian network</td>
<td>13, 24, 41</td>
<td>53, 55</td>
</tr>
<tr>
<td>EEG, see electroencephalography</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELAN</td>
<td>5, 6, 40, 41, 43, 44, 107, 108</td>
<td></td>
</tr>
<tr>
<td>electrode</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>electroencephalography, ii, 5, 106, 137, 138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elman network, 30, 33, see also neural network</td>
<td></td>
<td></td>
</tr>
<tr>
<td>English, ii, 11, 17, 18, 29, 39, 79, 81, 82, 84, 86, 87, 89–92, 95, 99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enron email corpus</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>ergodicity</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>event-related potential, ii, 5, 10, 43, 47, 106–108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>exchange algorithm</td>
<td>44, 68, 69</td>
<td></td>
</tr>
<tr>
<td>expectation maximization</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>feedback loop</td>
<td>135</td>
<td></td>
</tr>
<tr>
<td>filler word</td>
<td>55, 57, 95</td>
<td></td>
</tr>
<tr>
<td>Firth, J. R.</td>
<td>139</td>
<td></td>
</tr>
<tr>
<td>fish, 13, 39, 47, 52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fMLLR</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>fMRI, see functional magnetic resonance imaging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional magnetic resonance imaging, 138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>generalized iterative scaling</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>geometric mean</td>
<td>142</td>
<td></td>
</tr>
<tr>
<td>German, 10, 109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gigaword corpus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arabic, 79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>English, 2, 14, 34, 48, 60, 61, 68, 79, 110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good-Turing estimation</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>googol</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>hash table</td>
<td>68, 93</td>
<td></td>
</tr>
<tr>
<td>hidden layer</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>hidden Markov model</td>
<td>41–43, 45</td>
<td></td>
</tr>
<tr>
<td>hill, 107</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hippocampus</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>HMM, see hidden Markov model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HRWAC corpus</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>Hungarian, ii, 79, 81, 82, 84, 86, 89, 91–93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hurling</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>hyperbolic tangent</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>inferior frontal gyrus</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>information retrieval</td>
<td>139</td>
<td></td>
</tr>
<tr>
<td>interpolation</td>
<td>84, 86, 89, 99</td>
<td></td>
</tr>
<tr>
<td>ion channel</td>
<td>134, 135</td>
<td></td>
</tr>
<tr>
<td>iterated logarithmic growth</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Jaccard index</td>
<td>49, 142</td>
<td></td>
</tr>
<tr>
<td>Japanese</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>Jelinek-Mercer smoothing</td>
<td>14, 35, 36, 40, 55, 108</td>
<td></td>
</tr>
<tr>
<td>Joshua, 68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kaldi, 71, 96, 98, 100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Karen, 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Katz smoothing</td>
<td>14, 35</td>
<td></td>
</tr>
<tr>
<td>KenLM, 68, 111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Khmer, 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kneser-Ney smoothing</td>
<td>17, 27, 29, 30, 34, 66</td>
<td></td>
</tr>
<tr>
<td>modified, ii, 34, 37, 56, 57, 60, 62, 68, 82, 83, 85, 87, 88, 90, 97, 101–103, 108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Korean, 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kronecker delta</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

148
L-BFGS, 37
L2 cache, 93
latent semantic analysis, 23, 26, see also topic language model
probabilistic, 2, 38
Latin
   script, 81
lattice, 100–102
Levenshtein distance, 70
light verb, 107
log-likelihood ratio, 142
logical form, 39
LPC, 6
LVCSR, 96
machine translation, 2, 3, 12, 15, 30, 39, 63, 66–68, 76, 109, 110
magnetoencephalography, 138
Make, 69, 79
MapReduce, 2, 93
Markov assumption, 12, 13, 46, 87
maximum entropy language model, 35, 37, 38
maximum entropy model, 22
memory mapping, 111
middle temporal gyrus, 136
mkcls, 68
mmap, 111
MMN, 6
Model M, 37
modified Kneser-Ney, see Kneser-Ney smoothing
morphology, v, vi, 6, 62, 67, 93
Moses, 68, 80, 109
mountain, 107
N100, 6
N170, 6
N200, 6
N400, 3, 5, 6, 7, 40, 45–47, 67, 107, 108
Na+/K+-ATPase, 134
natural language generation, 39
near-infrared spectroscopy, 138
neocortex, 5, 137
neural network, 26
   feedforward, 26, 27, 38
   language model, 27
   recurrent, 27, 30, 31, 38
neuron, 134–136
neuroscience, 3, 48, 106
NIRS, see near-infrared spectroscopy
NIST, v, 15, 16, 34, 109
normalization, 26, 28, 38, 41, 49, 50, 93
νφ-LM, 40, 81–83, 85, 87, 90, 93, 97, 98, 100–104
odds, 142
one-hot encoding, 27
OpenMP, 68
OpenMT, v, 15, 109
optical character recognition, 12, 39, 110
P100, 6
P200, 6
P300, 6
### Document Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3a, 6</td>
<td>6</td>
</tr>
<tr>
<td>P3b, 6</td>
<td>6</td>
</tr>
<tr>
<td>P600, 5, 8, 8–11, 40, 62</td>
<td></td>
</tr>
<tr>
<td>parallelization, 2, 28, 35, 68, 94</td>
<td></td>
</tr>
<tr>
<td>parliamentary proceedings, 90</td>
<td></td>
</tr>
<tr>
<td>parsing, v, vi, 2, 8</td>
<td></td>
</tr>
<tr>
<td>part-of-speech, 19, 43</td>
<td></td>
</tr>
<tr>
<td>patent, 90</td>
<td></td>
</tr>
<tr>
<td>PCFG, <em>see</em> probabilistic context free grammar</td>
<td></td>
</tr>
<tr>
<td>Penn Treebank, 45</td>
<td></td>
</tr>
<tr>
<td>permutation, 39</td>
<td></td>
</tr>
<tr>
<td>perplexity, 20, 25, 29, 30, 34, 44, 49, 66, 68, 69, 69–75, 81–83, 85, 87, 88, 90, 97, 98, 102, 103, 105, 106</td>
<td></td>
</tr>
<tr>
<td>Persian, v, vi</td>
<td></td>
</tr>
<tr>
<td>PET, <em>see</em> positron emission tomography</td>
<td></td>
</tr>
<tr>
<td>pie, 107</td>
<td></td>
</tr>
<tr>
<td>plate notation, 23</td>
<td></td>
</tr>
<tr>
<td>PLSA, <em>see</em> latent semantic analysis</td>
<td></td>
</tr>
<tr>
<td>PMN, 5</td>
<td></td>
</tr>
<tr>
<td>pointwise mutual information, 142</td>
<td></td>
</tr>
<tr>
<td>Poisson-Stirling, 142</td>
<td></td>
</tr>
<tr>
<td>population coding, 137</td>
<td></td>
</tr>
<tr>
<td>positron emission tomography, 138</td>
<td></td>
</tr>
<tr>
<td>POSIX, 111</td>
<td></td>
</tr>
<tr>
<td>potassium ion, 134</td>
<td></td>
</tr>
<tr>
<td>probabilistic context free grammar, 2, 38</td>
<td></td>
</tr>
<tr>
<td>pruning, 71, 108</td>
<td></td>
</tr>
<tr>
<td>race condition, 28</td>
<td></td>
</tr>
<tr>
<td>rat, 136</td>
<td></td>
</tr>
<tr>
<td>repolarization, 136</td>
<td></td>
</tr>
<tr>
<td>resting potential, 134, 136</td>
<td></td>
</tr>
<tr>
<td>RNN, <em>see</em> neural network</td>
<td></td>
</tr>
<tr>
<td>sampling rate, 95</td>
<td></td>
</tr>
<tr>
<td>school, 55–58, 60, 62</td>
<td></td>
</tr>
<tr>
<td>semantic role label, 26</td>
<td></td>
</tr>
<tr>
<td>sequential consistency, 94</td>
<td></td>
</tr>
<tr>
<td>skip language model, ii, 19, 20, 38, 55</td>
<td></td>
</tr>
<tr>
<td>sodium ion, 134, 135</td>
<td></td>
</tr>
<tr>
<td>softmax, 28</td>
<td></td>
</tr>
<tr>
<td>software keyboard, 12, 39, 110, 111</td>
<td></td>
</tr>
<tr>
<td>Spanish, 78</td>
<td></td>
</tr>
<tr>
<td>speaker adaptive training, 96</td>
<td></td>
</tr>
<tr>
<td>speech recognition, ii, 2, 3, 12, 15, 19, 26, 27, 30, 38, 39, 63, 66, 67, 71, 96, 102, 106, 110</td>
<td></td>
</tr>
<tr>
<td>spellchecking, 110</td>
<td></td>
</tr>
<tr>
<td>stroke, 136</td>
<td></td>
</tr>
<tr>
<td>structured language model, 2, 39</td>
<td></td>
</tr>
<tr>
<td>subcorpus, 80</td>
<td></td>
</tr>
<tr>
<td>Switchboard corpus, 56–60, 62, 72, 95, 96–100, 102, 104</td>
<td></td>
</tr>
<tr>
<td>t-score, 142</td>
<td></td>
</tr>
<tr>
<td>teknonym, 60</td>
<td></td>
</tr>
<tr>
<td>telephone, 72, 95</td>
<td></td>
</tr>
<tr>
<td>topic language model, 23</td>
<td></td>
</tr>
<tr>
<td>transition model, 100</td>
<td></td>
</tr>
<tr>
<td>trie, 68, 111</td>
<td></td>
</tr>
<tr>
<td>trigger, 21, 22, 36</td>
<td></td>
</tr>
<tr>
<td>language model, ii, 21, 35, 38</td>
<td></td>
</tr>
<tr>
<td>trigram, 19, 22, 25, 26, 29, 35–37, 53, 55, 78, 87, 93</td>
<td></td>
</tr>
<tr>
<td>triphone, 96</td>
<td></td>
</tr>
<tr>
<td>tuba, 61</td>
<td></td>
</tr>
<tr>
<td>Tversky index, 142</td>
<td></td>
</tr>
<tr>
<td>typology, 90</td>
<td></td>
</tr>
<tr>
<td>Umm Kulthum, 60</td>
<td></td>
</tr>
</tbody>
</table>
unigram, 35, 36, 52, 57, 60, 78, 87
unigram rescaling, 25
Unix, 111
Urdu, 109
UTF-8, 81
Wall Street Journal corpus, 22, 34, 37, 72
witch, 78
Wizard of Oz, 134
WMT, see Workshop on Statistical Machine Translation
word error rate, 26, 27, 34, 35, 37, 70, 70–76, 96, 98, 100–105
Workshop on Statistical Machine Translation, 109
Yule’s coefficient, 49, 142
Z, see normalization